

Research Note 86-29

AD-A167 910

INTELLIGENT COMPUTER ASSISTED INSTRUCTION (ICAI):
FORMATIVE EVALUATION OF TWO SYSTEMS

Center for the Study of Evaluation
University of California, Los Angeles

Presidio of Monterey Field Unit
Jack H. Hiller, Chief

Training Research Laboratory
Seward Smith, Acting Director

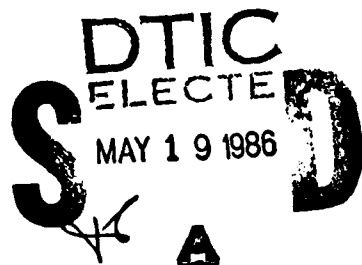


U. S. Army

Research Institute for the Behavioral and Social Sciences

March 1986

Approved for public release; distribution unlimited.



DTIC FILE COPY

86 5 16 08 8

U. S. ARMY RESEARCH INSTITUTE FOR THE BEHAVIORAL AND SOCIAL SCIENCES

A Field Operating Agency under the Jurisdiction of the
Deputy Chief of Staff for Personnel

EDGAR M. JOHNSON
Technical Director

WM. DARRYL HENDERSON
COL, IN
Commanding

Research accomplished under contract for
the Department of the Army

Center for the Study of Evaluation, University of California, Los Angeles
Jet Propulsion Laboratory

Technical review by

Ok-Choon Park
Joseph Psotka

Accession For	
NTIS GRA&I	<input checked="checked" type="checkbox"/>
DTIC TAB	<input type="checkbox"/>
Unannounced	<input type="checkbox"/>
Justification	
By	
Distribution/	
Availability Codes	
and/or	
Dist. Statement	
A-1	

This report, as submitted by the contractor, has been cleared for release to Defense Technical Information Center (DTIC) to comply with regulatory requirements. It has been given no primary distribution other than to DTIC and will be available only through DTIC or other reference services such as the National Technical Information Service (NTIS). The views, opinions, and/or findings contained in this report are those of the author(s) and should not be construed as an official Department of the Army position, policy, or decision, unless so designated by other official documentation.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER ARI Research Note 86-29	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) INTELLIGENT COMPUTER ASSISTED INSTRUCTION (ICAI): FORMATIVE EVALUATION OF TWO SYSTEMS		5. TYPE OF REPORT & PERIOD COVERED Final Report April 1984 - August 1985
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Center for the Study of Evaluation, University of California, Los Angeles		8. CONTRACT OR GRANT NUMBER(s) NAS7-918 Task Order No. 182/183 JPL Contract No. 956881
9. PERFORMING ORGANIZATION NAME AND ADDRESS Jet Propulsion Laboratory California Institute of Technology 4800 Oak Grove Drive, Pasadena, CA 91109		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 2Q263743A794 4413 102
11. CONTROLLING OFFICE NAME AND ADDRESS Army Research Institute for the Behavioral and Social Sciences. 5001 Eisenhower Avenue Alexandria, Virginia 22333-5600		12. REPORT DATE March 1986
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		13. NUMBER OF PAGES 358
		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE
16. DISTRIBUTION STATEMENT (of this Report) Approved for public release; distribution unlimited		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)		
18. SUPPLEMENTARY NOTES The contracting officer's representative was Jack H. Hiller.		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) Intelligent computer assisted instruction Intelligent tutoring Computer assisted instruction Artificial intelligence Formative evaluation		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) This report reviews major components of an 18 month evaluation of Intelligent Computer Assisted Instruction (ICAI), and emerging field of Artificial Intelligence that draws on computer technologies and cognitive science in an attempt to build more powerful instructional programs. The primary goals of this effort were to develop an increased understanding of the state of the art of ICAI for the purposes of: (a) identifying strategies to enhance the general usefulness of ICAI technology for Army training problems, and (b) developing concepts for efficiently and effectively managing military ICAI projects. (continued)		

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

i SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

item # 20 Abstract - continued

→ The approach taken to accomplish these goals was to intensively examine two selected ICAI systems using a formative evaluation methodology. The two systems selected were: (a) PROUST, a system designed by Soloway and Johnson for analyzing bugs in novice programmers' PASCAL programs, using a top-down approach which attempts to infer the intentions and plans of the programmer, and (b) WEST, a system designed by Burton and Brown to teach basic mathematics and strategic thinking skills, based on the premise that students can learn from their mistakes or "bugs". Formative evaluation is specifically designed to examine instructional processes and outcomes with particular attention to identifying programmatic strengths and weaknesses and formulating strategies for improvement. More specifically the studies focused on: (1) the product development cycle employed, including the institutional orientations of the designers and their sources of motivation; (2) the characteristics of the instructional strategies employed and the content addressed; and (3) empirical testing of the programs to determine what students learned and whether the program was more or less effective with certain types of students.

The report concludes with an analysis of lessons learned on the conduct of formative evaluations of emerging instructional technologies and strategies currently in use to develop ICAI programs. Suggestions are provided for increasing the effectiveness of ICAI products, for better aligning ICAI project design and military training needs, and for facilitating the management of future procurements and associated formative evaluation projects.

TABLE OF CONTENTS

I. Project Summary	1
Goals of the Project	1
Project Tasks	4
Accomplishment of Tasks	5
(Tasks 1-7)	
II. The Empirical Evaluation of ICAI Projects	9
What is ICAI?	9
Project Characteristics	10
Evaluation Focus	13
Evaluation Questions	15
References	16
III. Evaluation of PROUST	17
Chapter Summary	18
Introduction	19
Program Description	19
Evaluation Approach	20
Overview of Evaluation Methodology	21
Theoretical Orientation	21
Formative Review	23
Effectiveness Studies	23
Theoretical Analysis	25
Evaluation Questions	25
Theoretical Orientation	25
Model of Development	28
Formative Review	33
Evaluation Questions	33
Instructional Strategies and Principles	33
Overall Instructional Approach	33
ICAI and CAI Instructional Approaches	34
What PROUST Does	34
Quality of PROUST Content	42

Army Needs	42
Effectiveness of PROUST As An Instructional Intervention .	44
Evaluation Questions	44
Yale Study	45
Method	45
Results	55
Interpretation	67
UCLA Study	68
Method	68
Results	70
Technology Transfer	71
Conclusions from the Effectiveness Studies	72
Conclusions and Recommendations	73
References	76
IV. Shaping The Wind: Formative Evaluation of Intelligent Computer Assisted Instruction (ICIA)	77
Features of a Model for the Formative Evaluation of New Technologies	79
The Formative Evaluation of ICAI	83
Recommendations	91
References	92
V. Evaluation of WEST	93
Chapter Summary	94
Introduction	95
Literature Review	99
Theoretical Analysis and Formative Review	105
Evaluation Questions	105
Theoretical Orientation	107
Student Modeling	112
Knowledge Representation	116
Instructional Approach	118
Model of ICAI Development	131

Army Needs	131
Summary	133
Effectiveness Studies	134
Evaluation Questions	134
WEST Study 1	134
Method	134
Results	142
WEST Study 2	144
Method	144
Results	152
Interpretation of Effectiveness Studies	166
Conclusions and Recommendations	169
References	174
VI. Implications for Future ICAI Efforts	177
Improving the Instructional Development Cycle	178
Appendix A for: III. Evaluation of PROUST	180
Appendix B for: V. Evaluation of WEST	230

List of Tables for: III. Evaluation of PROUST

III-1. Instrumentation and Data Collection Strategy	22
III-2. PROUST and Instructional Development Characteristics	30
III-3. Contrast Between ICAI and CAI Systems.	35
III-4. PROUST Study Design at Yale (n)	47
III-5. Selected Variables by PROUST Treatment	48
III-6. Effectiveness Measures	50
III-7. Exam Results: Analysis of Variance	58
III-8. Exam Results: Means and SD	59
III-9. Ancillary Information.	61
III-10. N of Cases by Information Set	62
III-11. PROUST Questionnaire	66

List of Figures for: IV. Shaping the Wind: Formative Evaluation of
Intelligent Computer Assisted Instruction
(ICAI)

IV-1. Model of Relationships for the Formative Evaluation of New Technology	84
IV-2. Evaluation Model Information Flow	85

List of Tables for: V. Evaluation of WEST

V-1. Instrumentation and Data Collection Strategy	98
V-2. Means and Standard Deviations for Math and Strategy Measures.	153
V-3. ANOVAs for Group Effect, Pre-Treatment Differences	154
V-4. ANOVAs for Group Effect, Post-Treatment Differences	156
V-5. Performance Transcripts - Means and SD	157
V-6. Response Time Per Move Intercorrelations	159
V-7. Performance Transcript Correlations with Various Measures	160
V-8. Response Time vs Rating of Move	161
V-9. Means and ANOVA for Questions Asked	163
V-10. Correlations Between Math and Strategy Measures	164

List of Figures for: V. Evaluation of WEST

V-1. "Non-Constructive Error" Routine	109
V-2. Insufficient Instruction and Loss of Turn	111

(continued)

List of Figures for: V. (continued)

V-3. Lack of Effective Feedback	113
V-4. Lack of Entry Skills	115
V-5. Example of Coaching from WEST	120
V-6. Unidentified Issue	122
V-7. Excessive Bumping Causes Frustration	129
V-8. WEST Board Game	136
V-9. WEST 1 Math Key Card	137
V-10. WEST 2 Strategy Key Card	138

This work was performed for the Jet Propulsion Laboratory, California Institute of Technology, sponsored by the National Aeronautics and Space Administration.

Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not constitute or imply its endorsement by the United States Government, the Center for the Study of Evaluation, University of California at Los Angeles, or the Jet Propulsion Laboratory, California Institute of Technology.

I. PROJECT SUMMARY

This document presents a retrospective analysis of the major components of The Intelligent Computer-Assisted Instruction Study and a summary of its major activities and accomplishments. This introduction describes the major goals which guided the study and the tasks which were intended to accomplish these goals. A summary by task of accomplishments and problems encountered is then presented. Reports of the major study components are presented in subsequent sections. The report concludes with an analysis of lessons learned in this study that could benefit future R & D in the area of Intelligent Computer-Assisted Instruction (ICAI).

Goals of the Project

Over the last ten years, research efforts in the area of artificial intelligence (AI) has grown enormously and have addressed a number of different problems. Some work has sought to advance basic thinking in cognitive psychology by developing models of how people think; other research has taken a more problem-oriented focus and has examined alternative structures for developing knowledge bases, or what the computer "knows" and uses as an inferential data base. Recently, through the investment of Department of Defense resources, some R & D efforts have systematically attempted to use the power of artificial intelligence to help people learn through the use of Intelligent Computer-Assisted Instructional (ICAI) systems.

The roles these systems serve now, or could serve in the future, are various. Some systems operate by responding to particular queries that the student raises; others act in an associate or tutor role, diagnosing problems a student is encountering and providing instruction, information and/or feedback to help alleviate the problems. Yet, while these systems address instructional issues, it is unclear whether they incorporate principles of effective instruction and whether they in fact succeed as instructional strategies. In other words, how instructionally effective are current ICAI systems? Might they be made more effective? It is toward these general questions that this ICAI study was addressed.

To the extent that ICAI systems were designed principally as explorations of new technology, it is possible that explicit instructional principles were neglected, e.g., practice of increasingly complex material, feedback, prestructuring, and use of learning strategies. Thus, one purpose of the ICAI study was to review plans and actual system operations to determine the extent to which they used instructional principles.

A second weakness in the area of ICAI is documentation of the effectiveness of existing systems on learner performance. Because the technology is rapidly developing, success has often been regarded as sufficient when the system "runs." The ICAI study sought to examine, to the extent possible, the external validity of selected ICAI projects, that is, the effects these systems actually produced in learners.

A third area in need of examination was the documentary base for the R & D process in ICAI itself. Different clusters of researchers hold

radically different preferences in design strategies and do not communicate widely about their development processes. In order to facilitate the effectiveness of future design tasks involving ICAI, the study sought to document, on a case study basis for selected projects, the history of what was learned, what was tried, and what succeeded.

To these ends, the majority of resources for the study were allocated to the formative evaluation of representative ICAI systems to determine: 1) their instructional features; 2) estimates of their effectiveness; 3) documentation of their development history; and 4) suggestions for their improvement.

The project also pursued a second goal. What would ultimately be most useful in promoting effective ICAI applications is the development of an intelligent, computer-assisted instructional design aid, that is, a system which could help those responsible for training to prepare better instruction, training that is more comprehensive, efficient and effective. In order to approach plans for the development of such a system, the study sought to examine the state of the art in Automated Author-Aiding Systems, identifying critical and optimal features of such systems, and to analyze the implications of AI for future developments. The goal of this activity was to be a draft design for a first generation ICAI editor.

These, then, were the major goals for the ICAI study: to conduct a formative evaluation of existing models of ICAI; and to explore ways to harness AI in the service of better instructional design.

Project Tasks

In order to accomplish these goals, seven tasks were proposed:

Task 1 - Project Management, providing for the overall coordination, direction, control, reporting and reviewing of tasks 2 through 7 below;

Task 2 - Intelligent Computer Assisted Instruction Projects Survey Documentaton, providing documentation on references and information available on current ICAI projects and including an annotated bibliography, summary project descriptions, and an assessment of on-going development activities and future plans;

Task 3 - ICAI Projects Analysis and Evaluation Planning, providing for the selection of at least three (3) ICAI projects from those documented in Task 2 for detailed evaluation and the establishment of plans for evaluating each selected project;

Task 4 - ICAI Projects Evaluation, providing for the implementation of approved ICAI Project evaluation plans;

Task 5 - Automated Author-Aiding Systems Conference, providing for the planning and conduct of a conference to bring together experts in the field of Automated Author-Aiding Systems for the purpose of presenting and exchanging information related to this topic;

Task 6 - Design for Artificial Intelligence (AI) Editor, providing specification of requirements for developing an AI Editor to assist in instructional design;

Task 7 - Final Report, providing for the preparation of this document.

Accomplishment of Tasks

The above descriptions summarize project intentions, intentions which were accomplished in most cases. However, a number of problems emerged which influenced the timing and manner in which some tasks were pursued and, in one case, the accomplishment of the task itself. Task-by-task accomplishments are summarized below; problems are noted where they occurred.

Task 1 - Project Management. Because of occurrences described below, project management had to implement a number of changes in initial project plans, including modifications in scopes of work, budget reallocations and no-cost time extensions. These changes are discussed in relationship to the tasks they influenced.

Task 2 - Intelligent Computer-Assisted Instruction (ICAI) Projects Survey Documentation. This task was completed as planned and resulted in a report which included an annotated bibliography, the description of precontract visits to AI study sites, and documentation of rules of considering ICAI projects for study.¹

Task 3 - ICAI Project Analysis and Evaluation Planning. Key subtasks comprising this task included selecting ICAI projects for study and devising an evaluation plan for each selected project. This task was completed as planned, but its completion schedule was modified to accommodate necessary changes in projects selected for study.

Initial projects for study included Steamer, a Navy funded project

aimed at steamroom engineers; the Xerox maintenance tutor, funded jointly by ARI and Xerox; and PROUST, an ICAI program dealing with Pascal programming funded by ARI. Preliminary discussions suggested that each of the selected projects would be willing to participate. Evaluation planning commenced while more formal agreements were reached. Unfortunately, it was not possible to reach agreement with the Navy, and Steamer could not be included within our study. As a result, a new project was selected: WEST, an early but completed model of ICAI intended to help students acquire math and strategic skills. This needed substitution, as mentioned above, delayed the completion of this task and affected the timelines for subsequent tasks.

Task 4 - ICAI Project Evaluation. Several problems emerged in the conduct of the three evaluation studies. These problems modified the time schedule for the completion of this task and required the deletion of one evaluation study.

Communication was a continuing problem in the Xerox Maintenance Tutor Project. Although a primary coordinator at Xerox was assigned responsibility for coordinating evaluation-related activities and agreements were reached regarding sending CSE project documentation and including CSE in progress meetings with the Army, implementation of the agreements and access to Xerox was a continuing problem. Despite numerous followups, we were unable to get the information we needed to proceed with the evaluation or to arrange the necessary interviews with Xerox staff. As a result, the Xerox study was deleted from the scope of work. In its

place, an analysis of issues and problems in evaluating ICAI was completed. This analysis is included as Task 4B in this document.

While communication was a continuing problem with Xerox, the primary problem with the PROUST study was one of timing. The Yale developers of PROUST were very enthusiastic about collaboration, and expectations for UCLA and Yale contributions were clear. Unfortunately, however, Yale schedules of delivery were delayed somewhat from original plans, delays which significantly influenced UCLA operations. For example, the initial plan was to run parallel PROUST studies at Yale and at UCLA. However, delays in the transfer of PROUST files exacerbated technical problems in installing PROUST at UCLA. As a result, the UCLA trial had to be abandoned. In addition, the PROUST evaluation was originally intended to analyze a newly developed tutor; because this tutor was not completed, its analysis was impossible. The study investigating the PROUST program analyser was successfully completed and revised based on the review of project consultants. (See subsequent sections of this report.)

Task 5 - Automated Author-Aiding Systems Conference. This task was completed as planned. A conference bringing together government, private, and academic experts in the field was convened at UCLA in June, 1984 to synthesize the state-of-the-art in the use of computer-assisted instructional design. A report of that conference was submitted earlier in the contract.²

Task 6 - Design for Artificial Intelligence (AI) Editor. This task was intended to test the feasibility of an AI editor designed to create

multiple ICAI programs and to provide conceptual underpinnings for military planning efforts in this area. Delays in the contract noted above required no-cost time extensions for this aspect of the project as well. As a result, its completion was no longer considered useful to the military funders it was intended to serve. Consequently, this task was deleted from the scope of work.

Task 7 - Final Report. This document represents the completion of Task 7.

As is apparent in the summary above, the evaluation of the two ICAI projects, PROUST and WEST, was the principal focus of the contract. The following sections of this report describe these studies in detail, their results and implications.

Notes:

1. ICAI Projects Survey Report: Task 2 Deliverable, Intelligent Computer-Assisted Instruction Study. (July, 1984) Los Angeles: Center for the Study of Evaluation, UCLA.
2. Automated Author Aiding Systems Conference, Intelligent Computer-Assisted Instruction Study. (August 1, 1985) Los Angeles: Center for the Study of Evaluation, UCLA.

II. THE EMPIRICAL EVALUATION OF ICAI PROJECTS

Under contract to the Jet Propulsion Laboratory, an innovative effort was made to conduct a formative evaluation of intelligent computer assisted instruction (ICAI). This field is characterized by great activity, enormous promise, but, as yet, few strong working examples of instruction supported by the power of artificial intelligence based software. It was the intent of this project to describe, document, test, revise, retest, and provide recommendations for the improvement of model examples of ICAI. This work was to bring to bear the power of existing knowledge in the fields of instructional psychology and assessment so that the ultimate ICAI products might be more efficiently and effectively (intelligently) designed.

What is ICAI?

In order to evaluate ICAI, one must decide what the critical attributes of a system are. Rather than focusing purely on effects, that is what a system does, one must have a reasonably good understanding of the components to which successes and failures might be attributed. The problem is analogous to evaluating a human tutor. One could look at end measures to determine if students learn. However, in order to recommend ways to improve (which was a major emphasis of our study) one would wish to look at the domain knowledge (the content of what was taught), the teaching strategies employed, and the extent to which the tutor understood student problems and adapted accordingly.

ICAI attempts to parallel the above issues of domain knowledge, teaching strategies, and understanding of the student by including software components explicitly directed to each. Any ICAI system needs to include at least the following components (some ICAI systems do not have all three but for historical reasons have been classified as ICAI systems):

Knowledge representation. One of the central concerns in artificial intelligence research is the representation of real world knowledge in a computer. An intelligent tutoring system uses this representation of the content domain both to provide content for instruction and to evaluate the learner's responses.

Tutoring strategies. Separate from the knowledge-base is a control module which actually operates on the content. Rules for how to teach, when to help, how frequently and in what form, are incorporated into this component.

Student model. An intelligent tutoring system maintains a global perception of the learner. This component is updated by learner responses, provides diagnostic analysis, and is used to advise the tutoring strategy component of its next appropriate action.

Throughout our evaluation, these characteristics components of ICAI will be discussed as appropriate, in addition to the analytic and empirical efforts we undertook.

Project Characteristics

The examples selected for study were based upon availability (given the limited schedule for the evaluation) and cooperation by the program originators. Two ICAI projects were selected. One, WEST, an ICAI game

based on a Plato program, How the West Was Won, was used because it seemed to include desirable characteristics: a range of instructional goals; subject matter accessible to analysis; a clear instructional strategy. WEST is also an early example of ICAI and it was imagined that it could be contrasted with more recent efforts, to assess progress in the field. The second project, PROUST, consists at this point of an expert system for analyzing student PASCAL programs. Broadly speaking, it fits into the knowledge representation category above. It was planned that a more complete ICAI system an ICAI component would be available for analysis but PROUST project schedules did not permit such a test. An additional advantage of the contrasts between the WEST and PROUST projects was the difference in complexity of the content areas they were treating (arithmetic vs. computer programming) and the target populations (upper elementary age children and college freshmen.)

The selection of projects for this activity was not a trivial issue. The UCLA staff firmly confronted the basic problem of all evaluation efforts mounted externally to project development: skepticism and resistance to a process that seems directed to uncovering weaknesses and removing some level of project control from the designers of a technological innovation. A second problem, particularly exhibited in areas of rapidly expanding research and development, was related to the multiple purposes for which any project was undertaken. In particular, ICAI has not developed as an applied field where its actors single-mindedly, or even, emphatically, attempt to develop working, effective instructional implementations. Rather, the task of designing

operating ICAI systems functions as a rhetorical goal for a broad range of ICAI research and development efforts. The principal emphasis is on exploring the capabilities of AI on aspects of problems that would have to be solved were a functioning ICAI system designed. These problems involve the knowledge representation strategies, the requirements for natural language interfaces, diagnosing learner errors, and the rules by which problems are to be solved, among others.

Because ICAI efforts develop largely in a research rather than in a development context, certain facts characterize their activity. First, research goals, contributing to knowledge and theory building, appear to be paramount. Focusing on academically respectable efforts frequently characterise emerging, synthetic fields. (See for instance the spate of theory building in educational evaluation in the late sixties.) Second, efforts are selectively addressed based on the research predilections (rather than the project development requirements) of any particular set of investigators. Third, there are no real "shelf-item" components available for easy substitution into the project. Thus, if the researcher invests effort in knowledge representation, his final product may not work because of the lagged emphasis in another important component. The foreknowledge of uncertain success does not dampen the ICAI emphasis. First, the rhetorical quality of the goal should not be seen as misconducted marketing. In an emerging field, breakthroughs are anticipated. Secondly, keeping the idea, even as an idea, of ICAI futures in the mind of the researcher suggests fruitful paths of exploration.

Nonetheless, this analysis explains why the two projects in our study were selected. WEST, although created by Brown and Burton (1978) on a design by Bonnie Anderson (Dugdale & Kibbey, 1977), has only mild continuing interest on the part of its designers. They are less anxious about and even perhaps less interested in its current effectiveness, since the software is approximately ten years old and they have progressed to newer, more compelling problems. PROUST (Johnson & Soloway, 1983) was selected because its designer was one of the few in the field with documentation that suggested serious interest in whether the development of ICAI was effective with students, in addition to the research contribution he intended to pursue. These underlying orientations of ICAI development are critical to understanding the processes engaged in the evaluation, our interpretations, and the recommendations we provide for both designers and funders of ICAI efforts.

This report consists of two major sections: a report on the PROUST program and a report on the WEST program. In addition, a conceptual paper dealing in common language with the problems of ICAI development has been included.

Evaluation Focus

A three-phase evaluation template was designed for use in both project evaluations. The first phase of each evaluation included an attempt to understand the "product" development cycle employed, the ideological orientations of the designers, and their stated intentions. A second phase of analysis involved reviewing the internal characteristics of the ICAI systems from two perspectives: first, the quality of the instructional

strategies employed; and second, the quality of the content addressed. A third and major phase of the study was empirical testing of the programs. Here the intention was to document effects of the program with regard to individual difference variables among learners and with regard to a broadly conceived set of outcome measures including achievement and attitude instruments. Given optimal project schedules, the second intent was to modify the instructional conditions under which the ICAI system operated in the software is approximately ten years old and they have progressed to newer, more compelling problems. PROUST (Johnson & Soloway, 1983) was an attempt to make it more effective. Planned experimental comparisons were one option by which these instructional conditions could be contrasted. Based on these three major phases (theoretical, instructional, and empirical analyses), recommendations for the improvement of these particular projects and for the ICAI design and development process in general were to be developed.

The evaluation process also included other important features. First, a spirit of collaboration between the ICAI system designer and the evaluation team was to be developed, with frequent consulting on critical phases of the project as necessary. Second, a wide range of evaluation techniques were to be included, for instance, both quantitative and qualitative data collection and analyses.

What follows is the set of three evaluation questions addressed in both project reports and then the reports of the PROUST and WEST projects separately with conclusions and recommendations for future ICAI research and development.

Evaluation Questions

The evaluation questions guiding the study are presented below. In each of these, information related to the adequacy of the AI components (i.e., knowledge representation, instructional strategy, and student model) are treated as appropriate.

1. What is the underlying theoretical orientation of the system under evaluation? To what extent does the program serve as a model for ICAI?
2. What instructional strategies and principles are incorporated into the program? To what extent does the project exhibit instructional content and features potentially useful to future Army applications?
3. What are the learning outcomes for students? To what extent do learners achieve project goals? Do students with different background characteristics profit differentially from exposure to the project? To what extent does the program create unanticipated outcomes, either positive or negative?

Each of these questions will be applied to the two ICAI projects under review. The separate reports (as required in our contract) follow and may stand alone as separate papers.

REFERENCES

- Brown, J.S. & Burton, R.R. (1978) Diagnostic models for procedural bugs in basic mathematical skills. Cognitive Science, 2, 155-192.
- Dugdale, S. & Kibbey, D. (1977) Elementary Mathematics With PLATO. Urbana, IL: CBE Research Laboratory.
- Johnson, W.L. & Soloway, E. (1983) PROUST: Knowledge-based Program Understanding. New Haven: Yale U/CSD/PP #285.

III. EVALUATION OF PROUST

Intelligent Computer-Assisted
Instruction (ICAI) Study

CHAPTER SUMMARY

This report presents the description and evaluation of PROUST, an ICAI system designed by Soloway and Johnson for analyzing bugs in novice programmers' PASCAL programs. There are three major sections of this document: a theoretical analysis of the program, a formative review, and a report of two effectiveness studies conducted with PROUST. The purpose of this evaluation was to provide information relevant to the potential improved effectiveness of the system.

The theoretical orientation of PROUST is a top-down approach based on intentions and plans. Rather than compare the student program to an ideal implementation, PROUST compares it to the plan it believes the student was attempting. PROUST inspects a student's program and attempts to classify the inferred intentions against a set of possibilities based upon approaches of prior students. The program's greatest strength is perhaps its ability to deal with alternative goal decompositions.

PROUST is only a partial ICAI system in that it does not build a complete student model or have an expert tutoring system. However, two instructional features are germane to its bug analysis: feedback provided to students, and the timing of PROUST's use. Suggestions are made for improving the content, tone, and user-control of feedback. Additional recommendations are made for increasing the interactive aspects of PROUST's implementation through verification of student plans, input/output analysis, and student control of timing.

Effectiveness studies showed few significant findings related to learning outcomes. Students were generally positive about using the program although they had some criticisms. Difficulties of technology transfer are also addressed.

INTRODUCTION

This report describes the evaluation of an ICAI system entitled PROUST, designed by Soloway and Johnson at Yale University. The purpose of the evaluation was to provide information relevant to the potential improved effectiveness of the system. Throughout the process the UCLA staff were in regular contact with the program designers. Their cooperation was essential to the completion of this contractual task.

Program Description

PROUST is designed to assist novice programmers to use the PASCAL language in their own writing of computer programs. The approach taken is to provide intelligent feedback to beginning students about the quality of their efforts in an attempt to approximate the feedback that a human tutor might provide. In the words of its designers, PROUST is:

"...a tutoring system which helps novice programmers to learn to program"

"...a system which can be said to truly understand (buggy) novice programs," (Johnson & Soloway, 1983)

Thus, PROUST is not a trivial effort. In order to accomplish even this phase of system development, the designers have had to map the cognitive domain of computer programming, with PASCAL as the specific instance. The present implementation of PROUST permits students to submit their programs in response to two specific (but intended to be prototypic) programming problems. PROUST takes as its input programs which have passed through the PASCAL compiler and are syntactically correct. In analyzing these programs, PROUST attempts to infer students' intentions and to identify any

mistakes ("bugs") that occurred in the code (Johnson & Soloway, 1983). The system title is a literary allusion... "Remembrances of BUGs Past," with apologies to the original author.

As an example of a functioning ICAI system, PROUST represents only a partial solution. It contains the knowledge representation in software for the problemspace of the specific PASCAL programming problems. It also contains the diagnostic part of a tutoring component, which analyzes the student program to determine both student intentions and bugs. PROUST then provides feedback about its inferences about students' intentions and how well the student program implements the assumed plans. Currently under development is the pedagogical expert, which knows how to interact with and instruct (tutor) students effectively, and contains a student model to cumulatively monitor student progress. Although it had been anticipated that these components would be available for a full test of the ICAI system, schedule constraints restricted our activities to the completed components. The Yale project staff attempted to include an additional level of feedback in the analyzer as a precursor to the full development of the tutor. The findings of our study and recommendations related to instructional implementations will be considered as design and implementation of PROUST proceeds.

Evaluation Approach

In the evaluation of PROUST, three sets of questions guided our efforts. First, we wished to understand the orientation that the designers took and their development strategy. Here we were interested in the

theoretical underpinnings of the program design. Second, we were concerned with a formative review of the instructional approach taken with the system at its present level of implementation. Third, we were interested in the effects achieved by the program, both intended and unintended. The questions guiding the study are presented below:

1. What is the underlying theoretical orientation of PROUST? To what extent does PROUST serve as a model for ICAI?
2. What instructional strategies and principles are incorporated into PROUST? To what extent does PROUST exhibit instructional content and features potentially useful to future Army applications?
3. What are the learning outcomes for students? To what extent do learners achieve project goals? Do students with different background characteristics profit differentially from exposure to PROUST? To what extent does PROUST create unanticipated outcomes, either positive or negative?

Overview of Evaluation Study Methodology

Because the questions clearly call for a variety of data collection and analysis, ranging from review of documentation, inspection of the program, close observation of outputs from the programs, and student performance and self-report information, the procedures in the study were complex. Table 1 summarizes the instrumentation, data collection, and respondents required for aspects of the program under review.

Theoretical Orientation

Information in this area was collected through a review of project documents and through extended interactions with the Yale staff.

CSE's analyses and summaries were reviewed by consultants expert in AI, in PASCAL programming, and in instructional development. These analyses were shared with the Yale staff as possible.

Table III-1

Instrumentation and Data Collection Strategy

Evaluation Question	Dimensions of Inquiry	Measurement Method	Data Source
1. What is the underlying theoretical orientation of PROUST? To what extent does the project serve as a model of development for ICAI?	<ul style="list-style-type: none"> • theory of programming • cognitive underpinnings of programming • theoretical view of learning and instruction • ICAI development process 	content review interviews	primary documents project developers
2. What instructional strategies and principles are incorporated into the program? To what extent does the project exhibit instructional content and features potentially useful to future Army applications?	<ul style="list-style-type: none"> • instructional strategies and principles • subject matter content • Army needs 	program review	subject matter experts (instruction and PASCAL programming)
3. What are the learning outcomes for students? To what extent do learners achieve project goals? Do students with different background characteristics profit differentially from exposure to the project? To what extent does the program create unanticipated outcomes, either positive or negative?	<ul style="list-style-type: none"> • programming skills (bug identification and bug articulation) • background characteristics (academic history, computer-related experience) • intellectual self-confidence • reactions to PROUST • opinions toward computers, PASCAL programming • transportability of technology 	paper and pencil test questionnaire rating scale questionnaire opinion survey observation interviews	novice PASCAL programmers (college students) novice PASCAL programmers (college students) novice PASCAL programmers (college students) novice PASCAL programmers (college students) novice PASCAL programmers (college students) Technology transfer process

Formative Review

The formative review of instructional strategy necessarily focused on the feedback provided to the students by PROUST (the major instructionally oriented intervention.) This analysis was conducted at UCLA and resulted in a description and analysis of instructional strategies and principles used in the project. The findings were formulated based upon expert inspection of the program, the process of installing the program at UCLA (for a field test) and interviews with a number of beginning PASCAL students who submitted their programs to PROUST for analysis.

Effectiveness Studies

Effectiveness studies were designed to assess the levels of student attainment achieved as a function of PROUST exposure. Plans for two sets of studies were developed. One effort was conducted at Yale, the site of the system development. Here, contrasts in instance and numbers of problems and a No-PROUST condition were assessed. A second effort, full replication off-site at UCLA, was intended to assess the validity of the Yale studies and to assess the transportability of the system to another environment. The studies were intended to share certain features: 1) they were to be incorporated in regular course offerings as a credited part of university classes for non-computer science beginning programmers; 2) they were designed to use a wide range of measures, including newly developed criterion measures of programming performance, and attitude measures; 3) they were to include individual difference variables on students to determine if the PROUST experience was especially effective for particular subgroups of students.

The report will now revert to a structure related to the three major sets of evaluation questions. Processes and outcomes related to these questions will be described in turn, e.g., detailed procedures for the study designs will be treated in the effectiveness study section.

THEORETICAL ANALYSIS

Evaluation Questions

1. What is the underlying theoretical orientation of PROUST? To what extent does the project serve as a model of development for ICAI?

Theoretical Orientation

The system designers have as their purposes the creation of effective ICAI software and the necessary companion goal of providing a cognitive map of the programming domain. They have used the PASCAL language, which because of its structure, may provide one particular view of programming cognition, as opposed to the use of LISP or LOGO. Yet, they needed to start somewhere, and on a practicality basis, given the widespread offering of PASCAL instruction, their choice cannot be faulted.

The orientation of PROUST reflects the designers' theoretical approach to programming processes. PROUST'S developers see programming as a form of cognitive planning, in which the expert breaks the problem into segments, writes a plan for each segment, and then assembles the parts in a way to solve the given problem. Because experienced programmers will already often be familiar with requirements to implement segments of the plan, each solution may include a mixture of new segments and the application of routines in the repertoire of the programmer. The Yale staff believes that experts are more likely than students to use these synthetic approaches, and that novice programmers particularly have difficulty with analyzing their task into components; they are more likely to see the programming task as a monolithic one that requires an entirely new solution strategy.

Given the definitionally limited repertoires of novices, this is neither an unreasonable, nor unexpected course for them to pursue. This view of novice programming approaches is reflected in the underlying tack that PROUST takes (Johnson & Soloway, 1983 and 1985).

The approach based on intentions and plans (Schank & Abelson, 1977) is a top-down approach to programming, and as such has very subtle requirements for this project. The developers of PROUST started with the assumption that the analyzing and tutoring system should not only know what mistakes (or bugs) the students created but also needs to understand what the student's intentions were to assess their errors in the context of the particular approach the student was employing. According to the developers, the reason PROUST uses the intention based approach is that there is a high degree of variability in the data -- i.e., students can solve the same problem in a myriad of ways. There is no single or even small set of ideal models that would result in accurate program assessment. The developers feel it is THE major contribution to PROUST -- that PROUST can reason about alternative goal decompositions of a program, in addition to being able to reason about alternative implementation of a particular goal decomposition. Most other systems take the goal decomposition for granted and simply reason about alternative implementations of the given goal decomposition. However, in a domain such as computer programming, where there are many constructs that can be used, many ways to put them together, and many interpretations possible for an ostensibly clear problem statement, the developers found that there is

significant variability in the programs that students generate. Hence the need for PROUST's technique.

From a cognitive psychology perspective, this intention based approach is consistent with schema theory (Norman, 1976) that asserts for instruction to be meaningful it must be incorporated into students' current level and organization of information, and become elaborated as instruction proceeds. This model does not see the student as an empty vessel into which some black box set of information will be inserted. It is based upon the homely but powerful principle that learning must start where the learners are. Thus, PROUST is intended to harness the student schema by attempting to infer and recreate the student strategy, a process which could ultimately result in a very powerful instructional tool.

PROUST takes an intention-based approach to analyzing a novice's program. Rather than compare the student program to an ideal implementation (or solution to the problem) PROUST compares it to the plan it believes the student was attempting. PROUST inspects a student's program; it attempts to classify the intentions against a set of possibilities based upon prior student approaches. Given it has a certain level of confidence about what the student intention was, PROUST proceeds to identify bugs and provide feedback about the approach taken by the students. Because the system is constrained (it can't include every low probability "plan" that a beginning programmer might create), and it at present does not verify that it has inferred student intentions adequately, two sources of error exist. Nonetheless, when contrasted against an

approach that posits one or two "expert" solutions and matches student efforts against these limited templates, PROUST is an admirable attempt to break through the conformity that characterizes the direction of feedback students receive in beginning skill acquisition, as well as a serious effort to incorporate important tenets of cognitive psychology in this implementation.

Model of Development

The second issue in this evaluation question set is the extent to which PROUST serves as a model of development of ICAI. That question presupposes that ICAI projects are undertaken with a view to create, or to develop, a finished, effective product, a goal that was questioned in the introduction to this task report. Further, the term "model" of development needs some expansion. "Model" may be a set of regular and predictable relationships or may be thought of as the "city on the hill", the idealized process that activities should approximate. Last, "model" in the ICAI context could be normatively interpreted: Is this about the best of what the field does now, independent of one's assessment about its absolute quality?

Instructional development models have been in existence for forty years, (See Hovland, Lumsdaine and Sheffield, 1949 for reports on the "Why We Fight" series) and have various prescriptive characteristics. Stripping away the various ideological and semantic preferences in instructional development models, they tend to reduce to the following features: Does the design phase incorporate research knowledge (about learning,

instruction, etc.)? Is there a component orientation to design (making parts and combining them into systems)? Is there a commitment to collect process and outcome data on the level of effectiveness of the project? Are the data employed in systematic revision of the project?

How well does the PROUST project measure up on these points? Were we giving binary scores, PROUST would do rather well as a development effort.

However sunny the Table 2 marks, the character and quality of PROUST as a model of ICAI development have a long way to go. One precondition is that the designers decide to focus their major attention on the task of building an effective system. To do that, there would inevitably be a reduction on the research (or what computer science scholars term "theory-building") functions of their efforts. Because the work itself proceeds in no small measure with the energies of bright, dissertation oriented graduate students, a compelling impetus to continuing the research vs. development focus is undoubtedly present. But assuming this were a true development project (with all the unfairness of that assumption recognized), PROUST would be open to a different level of analysis.

Of particular note, and perhaps useful no matter what the distribution of project effort between development and research is the matter of data collection. PROUST staff have an adjunct project, funded by the Army Research Institute, for a relational data base designed to integrate process and outcome data on students. However, the quality of the outcome information used now could profit from some attention.

Table III-2

PROUST and Instructional Development Characteristics

FEATURE:	YES	NO
INCORPORATES RESEARCH KNOWLEDGE IN DESIGN PHASE	X	
COMPONENT ORIENTATION TO DEVELOPMENT	X	
COMMITMENT TO COLLECTING DATA:		
PROCESS	X	
OUTCOME	X	
USE OF DATA IN REVISING PROJECT	X	

First, there is the overall issue of what dependent measures might be useful. At present the course examinations serve this function (although considerable effort was made by the UCLA staff to modify these) and these examinations could stand revision. PROUST students are asked not to create programs (the nominal end in view of a programming class) but rather to adopt a PROUST persona and detect bugs in sample programs. They are given credit for those they find, and marked down for identifying bugs that don't exist. Furthermore, questions about the reliability and comprehensiveness of the scoring criteria for the analysis task can be raised. The preference for employing an analytic, error detection task rather than the synthetic requirement of building a program to solve a problem as the dependent measure needs attention. It is analogous in an eerie way to the focus on revision exercises in writing assessment, to avoid the costs, assumed low reliability, and effort required to acquire actual student writing samples.

Similar as well was the early focus on syntactic and mechanical errors (program syntax). The move, even within prose revision exercises, to semantic and organizational issues represented a precondition for the analysis of actual prose samples. Analysis may be a precursor to synthesis, it may be correlated with synthesis, but its cognitive demands are dramatically different. As PROUST moves toward tutor implementation, these differences need direct confrontation.

Second, the generalizability of stimuli to which students respond is a critical issue. PROUST works on two problems, with differing levels of

satisfactory analysis. To what problem space should this system generalize? Certainly, generalization versus limited focus has long been a point of contention in the AI community. Early attempts (Newell, Shaw & Simon, 1957; Ernst & Newell, 1969) pushed toward highly generalizable solutions. The pendulum has swung to microworlds of content parameters. How much time and energy (again if PROUST staff were to adopt a development goal) is it worth to create a system to analyze (and later to tutor) only two problems? If we were to hire a human tutor, we would rightly expect a wider repertoire. Thus the "prototype" function of the problems need review. Furthermore, "topic effects," what the content of problems are as opposed to the skills and processes they demand, have long been thorny psychometric issues in areas like science problem solving, prose analysis, and English composition. There is no reason to expect programming will be less impacted by this issue.

On the other hand, the PROUST repertoire is not limited when one considers the range of rules necessary to allow the system to consider a variety of student plans as the basis for analysis. PROUST could be expanded to consider even more novel plans, one assumes. But in assessing the quality of development, one might direct attention back to the range of problems, as well as of plans, that PROUST can reliably analyze.

FORMATIVE REVIEW

Evaluation Questions

2. What instructional strategies and principles are incorporated into the program? To what extent does the project exhibit instructional content and features potentially useful to future Army applications?

Instructional Strategies and Principles

In order to respond to the issues of instructional strategies and principles, we must address what appears to be an overall (and relatively common ICAI) approach to instruction. We then will describe in detail what PROUST does, how it is implemented in the context of regular instruction, and what reservations we have about its utility. In the course of this analysis we will also comment on the quality of content in the PROUST view of PASCAL.

Overall Instructional Approach

PROUST is based upon an instructional approach modeled on what human tutors do, that is, track student performance, diagnose sources of difficulty, and provide ways for students to ameliorate their problems. At this point, PROUST functions principally as a diagnostician, with the remedial, "how to fix" issues remaining for subsequent tutor implementation.

Tutorial approaches were thought to be most adaptive to individual difference and to best use the power of AI (Sleeman and Brown, 1982). Tutorial approaches can operate under two modes. They can attempt to teach

new knowledge or they can assume students have already had some exposure and thus focus effort on finding out what hasn't been learned. In both cases they attend to errors rather than to successes and can create problematic motivational effects. Moreover, there is still considerable difference among psychologists on the overall benefit to performance of building on strengths rather than identifying faults. Given this approach, ICAI systems eschew (in general) the provision of direct instruction on an issue, assuming experiences (often undefined) or insight (usually imagined) is the initial basis for student response. Thus, in PROUST we find the error orientation, especially true in programming which generated the bug metaphor now applied more generally to ICAI diagnostic efforts. We raise the point here, and will return to it in our recommendations regarding future PROUST development.

ICAI and CAI Instructional Approaches

Just so the context of the PROUST analysis is clear, in Table 3 we present a brief contrast between ICAI and CAI systems in terms of their typical instructional strategies.

What PROUST Does

Since PROUST analyzes and provides feedback to students in its current implementation, we will focus our comments on this area of instruction.

In its present form PROUST is restricted to a relatively static approach to interacting with students. The timing of PROUST analysis is restricted. For instance, students submit their draft programs to a PASCAL compiler, and then to PROUST. PROUST is not interactive. It passes the

Table III-3

Contrast Between ICAI and CAI Systems

Instructional Strategies and Principles	ICAI	CAI
Provision of instruction	student "discovered"	often direct
Character of responses	student generated	often selected or very restricted
Structure of instruction	based on student errors	often hierarchical part/whole
Tracking of student progress	cumulative student models	limited to previous hierarchy
Provision of feedback	frequent	frequent
Accuracy of feedback	comprehensive and adaptive	limited
Quality of content base	excellent	variable
Linkage to performance base	weak	strong
Potential levels of skills to be learned	enormous	limited by analysis of response options

students' programs through a pattern matching template and then informs students of bugs based on this analysis. PROUST analysis might be timed differently. Conceivably, students could move from the compiler phase to running their program (trying to debug on their own), then to PROUST. Here they would have the benefit of seeing their output and might be better able to understand PROUST's messages. Another way to approximate interaction even at this phase of PROUST development would be to provide student feedback on the plan PROUST infers the student used, then to allow the student the opportunity of fixing the plan, denying PROUST's plan, or in some way directly stating his/her intentions. Following this process, PROUST could analyze the student program.

PROUST might also be modified to cumulatively track students' successive submissions. In its present implementation, each submission to PROUST is treated as a one-time event, and no history of student efforts is maintained, should successive submissions and analyses be performed. We will expand on this student model in a later section. Certainly, the dynamic character of instruction will change with the creation of a functioning tutorial component, but these options may be considered at this point.

Looking specifically at the nature of feedback accuracy, there are at least three separate functions PROUST performs:

1. identifying plans (being able to analyze the problem and posit student intentions);
2. identifying bugs as bugs;
3. identifying correct elements as correct.

At this point, there is little independent evidence that PROUST correctly identifies student plans. There have been no reported comparisons, for instance, on whether PROUST has successfully inferred student intentions. According to the developers, in the strictest sense such information is not known. They would have had to run their data through other people's systems. The major problem with that technique is that there are no other systems that can handle programs of the complexity and variability that PROUST handles. Thus they could not run other tests. Inferentially, there are data that suggest that PROUST does a good job at bug identification, where PROUST has been demonstrated to analyze about 75% of the programs submitted to it and recognizes up to 95% of the bugs in the program (Johnson, 1985). How this compares with other methods of analyses, i.e., looking at input/output behavior, is unknown and perhaps unknowable. PROUST's creators have developed the code necessary to make use of I/O information and the comments to suggest that the student do some I/O testing. However, they decided not to use it at present since they felt they had no theoretical reason to think it would improve PROUST's performance dramatically and it was relatively untested. They point out that in software engineering environments I/O behavior is only one of many tests because it underdetermines bug cause. For example, an infinite loop can be caused by a very wide range of bugs. Moreover, when several bugs interact, as can often occur with novices' programs, I/O behavior can be misleading.

Nonetheless, although I/O analysis will not always indicate the origin of a bug, it can usually spot that there is a bug. As a human tutor might do, PROUST could first allow the student to see what his/her program does, and then look at the code to try to determine what is causing the problem. There are three reasons for advocating the addition of I/O analysis in a programming tutor: accuracy, relevance of feedback, and transference of skills learned. An extended example illustrating how the addition of I/O analysis to PROUST might enhance its accuracy, feedback relevance, and capacity to promote skill transference is included in the Appendix.¹

Our efforts at UCLA in replicating PROUST and our analysis of Yale PROUST protocols raise another question. When PROUST identifies false negatives (false alarms based upon faulty plan identification), what may be the instructional effects of such false alarms? Here we run directly into the technological authority of PROUST when compared with the insecure knowledge and confidence of students. If it were possible to make PROUST analyses more tentative, more maybes, et cetera, students could be encouraged to take the PROUST analysis with a grain of salt, and use PROUST bug identification as a cue to review and inspect their approach. On the other hand, that ability to challenge even a friendly and tentative _____

¹ From our study of PROUST it was necessary to make some inferences about how PROUST was used, including what constitutes correct input validation. Following review of a draft of this report by the Yale staff, they indicated that "correct input validation" was defined in class to include an error message. Thus the comments in the Appendix are not relevant to the Yale implementation of PROUST but should be relevant to implementations lacking this adjunct information.

computer program may require more confidence than beginning programmers (especially non-computer science majors) are likely to have, particularly because what they are learning is computer technology, not arithmetic or subtraction, areas in which the computer (or the instructors who designed the program) are not definitional experts.

Another concern with feedback is both a strength and a weakness. PROUST does comprehensive analyses based on its knowledge base. It gives feedback on minor errors as well as major ones. The volume of information it provides might be considered to be overwhelming or at least to raise questions about cognitive processing loads it lays upon novices.

An additional point might relate to the PROUST personality, earlier alluded to. Perhaps in the name of making the evaluation focus of PROUST analysis more palatable (and evaluation is difficult for all of us) PROUST might provide its messages in friendlier, more interesting ways. Again, if the purpose of the system is to help, perhaps PROUST can adopt a more kindly avuncular approach.

In considering future PROUST revisions, we would like to return to the issue of a complete student model. An important aspect of most intelligent tutoring systems is the model of the learner which it develops. Such a system bases any tutoring decision on an ongoing perception of the learner. The present implementation of PROUST creates a model of the program, but has no ongoing model of the learner. Every time it looks at a new program, it is as if it had never encountered the learner before. If a student submitted the same program twice, the two analyses would be identical.

A human tutor looking at a student's second (or subsequent attempt) will often ask, "What did you change this time?" This not only helps the tutor focus on how well the student has implemented his/her previous suggestions, it also gives him/her further insight into what the student is trying to do. This allows the tutor to use less directive feedback. If subsequent attempts show that the bugs remain, the tutor can be progressively more directive. Similarly, when the tutor has ascertained that the student has attained some proficiency, the feedback can gradually fade so that the student learns to work on his/her own.

Thus, it would be desirable for the pedagogical expert to add another level of modeling above the programming expert. The pedagogical model would integrate information from the programming model to form a more complete picture of the learner. This information could in turn be used by the present control structure to help determine the learner's strategy. One of PROUST's strengths is its ability to spot small bugs which would not show up in an I/O analysis because they are blocked by larger problems. While this information should be added to the top-level model of the learner, it is perhaps not relevant to the learner until he/she solves the major bug and gets the program running. Again, because the pedagogical expert will not be required to provide complete tutoring in one shot, there will be more of an opportunity to facilitate the learner's skill development.

To illustrate the type of desirable interaction, an idealized dialogue for an actual sequence of programs has been written and is included in the

appendix. The programs come from a protocol provided by Yale with PROUST's output and annotation provided by Soloway. Following each program and analysis are the idealized pedagogical expert's replies to the student.

The idealized replies may be unrealistic given present technology. However, attempts were made to avoid interaction between the student and the tutor due to present limitations in natural language understanding. There is one point at which the ideal tutor asks the learner a question, but the tutor is only looking for a simple yes or no.

In summary, a proper instructional analysis of PROUST is difficult since by the authors' description it is a bug analyzer, not an instructional system. Nonetheless, PROUST does have two important instructional features that merit comment: the feedback provided and the timing of PROUST's use.

PROUST's bug identification feedback is detailed and fairly comprehensive, attending first to major bugs and then to minor ones. However, some students need to know more than just what is wrong. They need more instructive feedback that tells them how to fix their mistakes, possibly graduated from more to less directive over time. The feedback is accurate in finding bugs but occasionally gives "false alarms" in a way that may convince novices that they are wrong when they are actually right. Verification of student plans rather than complete reliance on inference might increase the power of the system.

Student control of when PROUST is used would also be likely to increase the effectiveness of the system. For example, some students are

overwhelmed by receiving all bug feedback at one time. In addition, many students might benefit from seeing what happens when their program runs before they submit it to PROUST for bug analysis. The Yale staff purposely introduced PROUST at a fixed point (after compiling and before running the programs) in order to reduce variability in the experimental setting. However, in a normal instructional setting, we would hope students could use it on whatever basis was most appropriate to them.

Quality of PROUST Content

This section is mercifully brief. In the context of the formative review of PROUST we submitted the PROUST problems (for the course final examination) project documentation, and student protocols to analysis by experts in the PASCAL programming language. These included two UCLA professors in computer science, two UCLA mathematics department professors who taught introductory computing, a UCLA psychologist responsible for teaching introductory PASCAL, and an advanced student pursuing graduate work jointly in AI and educational psychology. None of these individuals had any serious reservations with the quality of the PASCAL content addressed. They believed the problems presented and character of feedback were appropriate for the content area under development. Some preferences for further system documentation were expressed, however.

Army Needs

To what extent does the project exhibit instructional content and features potentially useful to future Army applications?

Assessing the utility of the instructional procedure for future Army needs is a difficult task, particularly because the full tutor implementation is not available for review. But in a general sense, certain comments can be made. First, the military must continue to pursue technological improvements to training and instruction as it is faced with increased complexity of new systems and fewer resources (people, time and money) to accomplish goals. It is necessary to make training accessible in the relative absence of experts in the latest technology; hence the need to rely on intelligence within the technology itself. Second, the use of systems that have at their heart the ability to adapt to student responses and to provide differential feedback will remain important as the Army increasingly reflects the growing cultural diversity of the United States. Spending resources to figure out how conceptual areas in technology (like PASCAL) are developed will undoubtedly shorten the development cycle for future projects of this sort. Are there any specific features of the PROUST instruction that can be recommended for general review and implementation for the Army at this time? Not at present. May there be? Certainly. Can PROUST itself be useful to the Army? Yes, if a number of conditions exist. Certainly, the Army needs to train technical people in computer languages. If PASCAL is in that language set, then this system could be useful if it were accompanied with standard direct instruction to provide the information and skills that students at Yale receive through courses and if it were simplified to reflect Army personnel ability levels, undoubtedly lower than entering Yale freshman.

EFFECTIVENESS OF PROUST AS AN INSTRUCTIONAL INTERVENTION

Evaluation Questions

3. What are the learning outcomes for students? To what extent do learners achieve project goals? Do students with different background characteristics profit differentially from exposure to the project? To what extent does the program create unanticipated outcomes, either positive or negative?

The goal of this aspect of the study was to conduct empirical studies of PROUST effectiveness. Here our intent was to examine under experimental contrasts the effects of PROUST on a range of student measures. In addition to performance in the PASCAL programming area, we were also interested in student reactions and attitudes related to the intervention. Our plan was to conduct two separate sets of studies of beginning PASCAL students, one conducted at Yale and the other at UCLA. We would expect, for instance, the Yale study to yield somewhat more positive results than the UCLA study for the simple reason that Yale was the development site, and staff there were substantially more experienced in PROUST application. The goal at UCLA was similar. However, we expected that we would confront issues related to transportability of the system, both in terms of its fit conceptually into programming classes at another major university, and any technical problems in getting the system to work in a foreign computer environment. Also the UCLA replication would permit the detached, controlled analysis usually thought to be a benefit of off-site

replication. The UCLA study had the additional merit of allowing UCLA staff access to the process that PROUST used with students while keeping travel costs low. Both experiments were intended to function within normal course environments rather than as additions or clear laboratory like sessions. For one thing, student effort was expected to be more seriously exerted in the context of regular course environment. It was felt that the verisimilitude that course embedding would provide would greatly outweigh the potential schedule and contamination effects of regularly instructional offerings.

Yale Study

Method

Design

The effectiveness study at Yale compared the two groups of students in a value-added evaluation design. One group was allowed to use PROUST for the two of their eight homework assignments which it can analyze. The second group was not given the opportunity to use PROUST. Thus, the issue examined was the value of adding PROUST, admittedly a short treatment, to the standard course instruction. It should also be noted that due to a lack of graduate teaching assistants for grading assignments, homework problems were made optional. Students were encouraged but not required to show their work to undergraduate students. Thus a second factor entered into the study design: student choice whether to do the assignments. Since there were two possible assignments for students to complete (the "Rain" and the "Bank" problems described later) there were four possible levels of homework actually accomplished for subjects in each group:

- . doing both the Rain and Bank problems
- . doing the Rain problem only
- . doing the Bank problem only
- . doing neither the Rain nor the Bank problem

Thus, the final design of the study was a 2 x 4 factorial design of opportunity to use PROUST by homework assignments completed. Table 4 shows the number of Yale subjects in each cell of the design.

Subjects

The study at Yale involved a total of 141 students enrolled in two identical sections of a beginning PASCAL programming course for non-majors. Both classes were taught by the same instructor, both were open to the same students, and both used the same methods, materials, homework and tests. The classes were offered in the spring semester of 1985. Although it was preferred, students were not randomly assigned to treatment groups. One classroom was given the PROUST opportunity and the other was not. According to Yale faculty, there was no reason to assume any systematic differences between the two groups of students. Thus the unit of analysis was considered the students rather than the classroom.

To verify assumed equivalence between sections, t-tests were conducted on selected variables, including general ability indicators such as the SAT and GPA, psychological indicators and exam scores. No significant differences were found, as shown in Table 5. This table also reports class composition by average score on several measures. Of the 69 students in the PROUST class, 36 were male, 33 female. Of the 72 students in the no-PROUST class, 37 were male, 35 female.

Table III-4

PROUST Study Design at Yale (n)					
Opportunity	Homework Turned In				Total
	Rain Only	Bank Only	Rain + Bank	None	
PROUST	12	2	19	36	69
No PROUST	15	4	19	34	72
Total	27	6	38	70	141

Table III-5

Selected Variables By PROUST Treatment

Variable	Opportunity	Mean	s.d.	t	p
SAT - verbal	No-PROUST	662.29	69.67	0.66	n.s.
	PROUST	652.88	80.26		
SAT - math	No-PROUST	648.36	73.08	0.81	n.s.
	PROUST	637.20	72.14		
GPA - college	No-PROUST	3.30	0.44	0.57	n.s.
	PROUST	3.25	0.38		
Intellectual Self-confidence	No-PROUST	142.49	5.67	0.52	n.s.
	PROUST	141.92	6.40		
Computer Attitude	No-PROUST	130.12	17.55	0.66	n.s.
	PROUST	132.67	16.84		
Midterm exam total score	No-PROUST	54.22	31.12	0.21	n.s.
	PROUST	53.04	34.74		
Final exam total score	No PROUST	58.38	26.38	0.21	n.s.
	PROUST	59.44	30.60		

Measures

Several measures were used in this study of PROUST's effectiveness. Copies of these are appended following the discussion section. Table 6 summarizes the content and timing of the measures.

Student Information Form. This form was administered at the beginning of the course and consisted of two parts. The first part was the Background Information measure, a self-report questionnaire designed by CSE staff, which elicited students' SAT verbal and math scores, high school and college grade point average, number of high school and college math and science courses taken, declared college major, and 15 items on computer background. It was administered at the beginning of the course. The second portion of the Student Information form consisted of the Intellectual Self-Confidence Inventory (ISCI). This scale, developed by Hiller (1974), consists of 30 items arranged on a 4 point scale from strongly agree to strongly disagree. A typical item follows.

4. It would be accurate to say that I really enjoy toying with ideas.

1	2	3	4
strongly agree	agree	disagree	strongly disagree

The ISCI scale was titled "Personal Outlook Inventory" when given to students as part of the Student Information Form.

Programming Performance Measures. CSE staff undertook extensive efforts to develop measures appropriate for the PROUST evaluation. PASCAL topic areas and skills were identified, and alternative measurement strategies, including student production of solutions to programming

Table III-6

Effectiveness Measures		
Measure	Content	Timing
Student Information Form	Background information,	Beginning of
2 subscales:	e.g. SAT scores, GPA,	Spring term
1. Background Information (all Ss)	number of math/science courses, computer experience.	
2. Intellectual Self- Confidence Inventory (given to all Ss; termed Personal Outlook Inventory when given to Ss.)	30 items reflecting self-confidence related to ideas & exposition.	
Midterm exam	3 problems in which Ss identify & explain bugs in written program	Middle of term, prior to Spring Break
Student Questionnaire		Immediately after Spring Break
2 subscales:		
1. PROUST Questionnaire (for PROUST Ss only)	16 questions on Ss' attitudes toward using PROUST	
2. Computer Attitude Scale (for all Ss)	11 questions on Ss' experience with PASCAL during course, and 30 questions about Ss' general computer attitudes and anxieties	
Final exam	2 problems in which Ss identify & explain bugs in written programs	End of term

problems were designed. Extensive telephone interaction occurred with the Yale staff related to the use of these measures. In the end Yale used the measures it designed; however, they reported that they expended considerable effort designing test instruments and incorporated a number of aspects of the UCLA measurement strategy. Drafts of the topic analyses and measure specifications attempted by UCLA are appended to this report.

The Yale Midterm was an hour-long examination developed to assess how well students had mastered the issues and skills treated in the course. It consisted of three paper and pencil problems where the student were to identify bugs in written computer programs and to explain why they were bugs. The students' role was much like PROUST.

The three problems were judged to be conceptually isomorphic to the two PROUST homework problems (Rain and Bank); they used the same algorithm but in different contexts. The bugs employed were those which PROUST is good at identifying, and which students usually have trouble finding, e.g., protecting against no inputs. For the three problems there were a total of 15 pairs of items (identifying and explaining), resulting in a 30 item measure.

The Yale Final was very similar to the midterm and consisted of two problems with the same type of bugs and requiring the same skills, (identifying and explaining bugs). It was judged by the instructor to be somewhat harder than the midterm. There were 11 pairs of items, i.e., 22 total items on the final.

Because the measures did not directly deal with constructing programs, they can be assumed to detect only two major characteristics of performance. First, they indicate whether students have mastered a necessary subskill of programming, noticing and explaining errors. Secondly, they may detect the extent to which PROUST served as a model of this process.

One area of uncertainty is the quality of the criteria used to classify students' responses. We were not provided with clear criteria for scoring employed by Yale staff so we were unable to independently score them as planned. Training processes are similarly unclear. We would have wished to know whether student efforts were double-scored to assess interrater agreement (and at what level), what the reliability levels were that obviated this requirement, and what training routines were necessary, if any, to prepare scores for valid and reliable assessment.¹

Student Questionnaire. The Student Questionnaire was given immediately after the spring break which followed the midterm. It consists of two portions. (1) The PROUST Questionnaire portion was given only to students in the PROUST group. It consisted of 16 items gathering attitudinal reactions to PROUST and computer programming. Below is an example of these items.

¹ Following review of the draft report, Yale staff informed us that resources had not been available for double scoring there. According to Soloway, "What can be said about our grading criteria is that we exercised care and that we discussed the range of answers as a group, and came to a consensus as to what should be learned."

5. Would you recommend PROUST to other students?

Yes _____ No _____

Why? _____

(2) The Computer Attitude Scale portion of the Student Questionnaire was given to students in both the PROUST and no-PROUST groups and was used to gather data on students' attitudes and anxiety towards computers. The first 11 items were on a 5-point and scale related to students' experience with PASCAL, e.g.

5. How easy has it been for you to learn to write computer programs in PASCAL?

1	2	3	4	5
very easy	fairly easy	about average	fairly hard	very hard

The next 30 items of the Computer Attitude Scale were developed by Gressard and Loyd (1984) and dealt with computer attitudes and anxieties. Items were arranged on a 4-point scale of strongly agree to strongly disagree, e.g.,

1. Computers do not scare me at all.

Strongly Agree	Slightly Agree	Slightly Disagree	Strongly Disagree
()	()	()	()

Procedure

At the beginning of the semester all students were asked to fill out the Student Information Form which included the Background Information and the Intellectual Self-Confidence Inventory. During the 13 weeks of the

course, eight assignments were given, the fourth being Rain, and fifth being Bank. Students in the PROUST group were given the opportunity to use PROUST on those two homework assignments. The day after Bank was due, the midterm was given. After the midterm and spring break students were given the Student Questionnaire, which included the PROUST Questionnaire for students in the PROUST group only, and the Computer Attitude Scale for all students. The final exam was given at the end of the course.

Homework assignments were similar to what was done in class but were set in a new context. Students had to be able to generalize what was done in class and transform it to fit the assignment. After working out a program either on- or off-line, students typed their programs into the computer. The compiler checked them and rejected those with syntax errors. For students in the PROUST group, once their programs compiled correctly, the programs were automatically submitted to PROUST for bug analysis. Students in the non-PROUST group bypassed the PROUST step. If there were any problems with the programs, the students were to revise them and then resubmit them to the compiler and PROUST. Throughout the process all students in both groups had access to teaching assistants.

The Rain and Bank assignments were as follows:

Rain:

Noah needs to keep track of rainfall in the New Haven area in order to determine when to launch his ark. Write a program which he can use to do this. Your program should read the rainfall for each day, stopping when Noah types @qt"99999", which is not a data value, but a sentinel indicating the end of input. If the user types in a negative value the program

should reject it, since negative rainfall is not possible. Your program should print out the number of valid days typed in, the number of rainy days, the average rainfall per day over the period, and the maximum amount of rainfall that fell on any one day.

Bank:

Write a Pascal program that processes three types of bank transactions: withdrawals, deposits, and a special transaction that says no more transactions are to follow. Your program should start by asking the user to input his/her account ID and his/her initial balance. Then your program should prompt the user to input the transaction type.

If the transaction type is an END-PROCESSING transaction the program should print out the (a) final balance of the user's account, (b) the total number of transactions, (c) total number of each type of transaction, and (d) the total amount of the service charges, and stop.

If it is a DEPOSIT or a WITHDRAWAL, the program should ask for the amount of the transaction and then post it appropriately.

Use a variable of type CHAR to encode the transaction types. To encourage saving, charge the user 20 cents per withdrawal, but nothing for a deposit.

Results

Midterm. Students (N=141) taking the midterm achieved a mean score of 53.65 (sd. 32.83), with a range of -27 to 120, or 147 points, about 55% on average of the best obtained score. The number of possible points was 150. Sixty-nine students availed themselves of the opportunity to submit their programs to PROUST, and averaged 53.04 (s.d. 34.74). The control group's (No-PROUST opportunity) average score was 54.22 (s.d. 31.12). The highest score obtained under the PROUST condition was only a few points

below the maximum. Negative scores were obtained by one or more students in every program condition. Reliability analysis for the items within the midterm showed that the items are reliable (coefficient alpha = 0.785). Only four of the items are relatively poor by conventional psychometric standards. The items constitute pairs of entries; the average intercorrelation among item pairs was 0.909, suggesting that the elements within a pair are mutually redundant.

Final examination. The final examination was administered to 129 of the 141 students present at the midterm. Mean total score for all students was 58.88 (sd 28.35), and the scores ranged between -10 and 120. Average percentage performance against the obtained range was 53%. A total of 120 points was available, and average percentage against the maximum possible was 53%.

The correlation between midterm and final total score was 0.586 (figured for PROUST and no-PROUST students together since there was no significant difference between them) suggesting only a moderate relationship between the two exam scores. On the final there were no significant differences observed between the experimental and control groups. Reliability analysis of the final exam items suggest that the exam (and the rater) is a highly reliable instrument (coefficient alpha = 0.875). The correlation between related pairs of entries is 0.930. Only two items show relatively low item-total correlations; however, their removal would add only slightly to the overall reliability figure and may be inappropriate because of content representation considerations.

When the results are inspected with regard to which homework assignments were completed (with the options being both Rain and Bank, Rain only, Bank only or neither) the findings take on some additional meaning. As shown in Table 7, on both the midterm and final a main effect was found for students who did not complete homework assignments (independent of PROUST exposure). Significant differences (Midterm: $F=10.21$, MS error = 972.65, $p<.001$; Final: $F=9.08$, MS error = 739.16, $p<.001$) were attributable to the fact that students who did not turn in homework averaged about one-third lower scores than those who did complete assignments.

When one inspects Table 8, which presents the performance measure results, one is struck by the relatively low numbers of students completing homework in either group; close to half the students in both experimental and control conditions chose not to do homework, and their scores on the midterm and final suffered as a consequence.

Notice that these students dropped out of the course at a marginally higher rate (as judged by changes in number from midterm to final completion). As the table indicates, the Bank only condition was dropped because of the few and therefore unreliable data provided in this condition.

The distribution of homework completion is readily explainable. About half the students completed the first homework assignment, RAIN. About two thirds of those completed both the Rain and Bank problems, and these findings are consistent across treatments. Essentially, one could interpret these figures to mean that exposure to PROUST neither motivated

Table III-7

Examination results: Analysis of variance

Midterm exam

	SS	df	MS	F	p
PROUST	0.59	1	0.59	0.00	n.s.
Homework	19854.89	2	9927.49	10.21	< .001
PROUST * Homework	686.88	2	343.44	0.35	n.s.
Error	125472.45	129	972.65		

Final exam

	SS	df	MS	F	p
PROUST	7.05	1	7.05	0.01	n.s.
Homework	13423.92	2	6711.96	9.08	< .001
PROUST * Homework	1014.09	2	507.04	0.69	n.s.
Error	87220.36	118	739.16		

Table III-8

Means and SD

Examination results							
Opportunity	Homework	Midterm			Final		
		n	mean	s.d.	n	mean	s.d.
Received PROUST							
	Rain	12	61.75	39.71	11	49.00	26.69
	Bank ¹	2	22.00	45.25	1	48.00	-
	Rain + Bank	19	70.53	26.15	19	79.32	25.89
	None ²	36	42.64	32.87	30	51.07	29.97
	Overall	69	53.04	34.74	61	59.44	30.60
Did not receive PROUST							
	Rain	15	52.40	34.17	15	53.67	31.89
	Bank ¹	4	56.25	16.26	4	46.00	11.19
	Rain + Bank	19	72.89	23.06	19	70.74	20.44
	None ²	34	44.35	31.19	30	54.57	26.41
	Overall	72	54.22	31.12	68	58.38	26.38
Item reliability							
			0.78			0.87	

¹ "Bank" homework results deleted from further analyses

² Did not do the homework assignments

nor discouraged students differentially from the next homework task. One potential factor in low Bank problem completion was the proximity of its due date to the midterm, which was probably a higher concern for students, given the situation in which homework isn't graded and tests are.

Table 9 presents the results arrayed by background characteristics of SAT performance and college GPA, as well as scores on the Intellectual Self Confidence Scale and Computer Attitude. The decrement in numbers by condition reported in this table is explained by incomplete data depicted in Table 10.

Academic Standing. The Yale students, as one would expect, are a highly talented group. For the group as a whole (including those who did the Bank assignment omitted from Table 9) SAT scores average 658.05 and 643.33 for verbal and math scales respectively, with scores as high as 800 and 790. About ten percent of the group reported twin 700+ scores. High school GPA averaged 3.77, and college GPAs averaged 3.28. The SAT math score is uncorrelated with any of the performance indicators; the SAT verbal score is significantly correlated with both midterm and final examination ($r=.54$ and $.59$ respectively.) This suggests that the verbal content of the problems may have a significant part to play in student performance.

Most of the declared majors were in liberal arts. Very few were in the hard sciences (the class was designed for non-majors). Major concentration had no bearing on the students' performance in terms of either examination. A similar lack of relationship is found with the

Table III-9

Ancillary information						
Opportunity	Homework ¹	SAT		College	Intellectual	Computer
		Verbal	Math	GPA	Self-Confidence	Attitude
Received Proust						
	Rain:	n	8	8	8	7
		mean	642.50	672.50	3.40	144.38
		s.d.	75.73	43.67	0.25	6.05
	Rain + Bank:	n	16	16	16	12
		mean	674.00	634.38	3.46	140.81
		s.d.	85.90	84.38	0.32	6.53
	None:	n	24	24	21	25
		mean	645.83	621.25	3.04	142.00
		s.d.	80.05	66.94	0.36	6.60
Did not receive Proust						
	Rain:	n	14	14	10	14
		mean	645.71	659.29	3.29	144.07
		s.d.	83.46	78.59	0.52	4.94
	Rain + Bank:	n	17	17	12	19
		mean	683.53	662.35	3.51	143.53
		s.d.	57.33	56.07	0.26	6.00
	None:	n	28	28	25	33
		mean	656.79	635.71	3.19	141.33
		s.d.	70.50	81.17	0.46	5.84
Range of data	Highest value		800	790	4.00	159.00
	Lowest value		430	430	2.00	125.00

¹ "Bank" homework results deleted from above analysis.

Table III-10

N of Cases by Information Set

Information set	<div>CompleteIncomplete</div>													Total N
Midterm Exam	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	141
Final Exam	✓	✓		✓		✓	✓	✓		✓	✓			129
Intellectual Self-Confidence Inv.	✓	✓	✓	✓	✓	✓		✓	✓				✓	120
Student Info. Form: background	✓	✓	✓	✓	✓				✓					111
Computer Attitude Scale	✓	✓	✓		✓	✓	✓				✓			83
PROUST Questionnaire	✓		✓				✓							30
	N	20	43	5	39	2	5	5	3	2	3	11	1	2

NOTE: Table 10 shows the distribution of respondents within each of the six sets of information gathered and provides an overview of attrition in this study. Because participation was voluntary, one or more pieces were missing for 86% of the students. The columns of Table 10 are arranged in descending order by number of participants completing a dataset; rows are arranged by descending order of number of information sets completed by a given group of participants. For example, 129 students took the final. Of these, 20 students provided information on all six measures, 43 more supplied information on all measures except the PROUST questionnaire, 39 others supplied information on only the midterm, final, ISCI, and Student Background, etc.

number of high school and college mathematics courses already taken. However, an exception occurs at the extremes: students with a large number of such courses tend to score higher on both exams than students with very few such courses.

Intellectual Self-Confidence Inventory (ISCI). The ISCI was administered to 120 students. Item reliability for this instrument is only moderate (coefficient alpha = 0.63), the individual scale items generally weakly correlated, and several of them could be deleted in order to add slightly to the overall reliability. The high level of functioning (as self-reported) is demonstrated by the elevated mean (142.25 of 144 possible) and low variability (sd = 5.97.) The scores on the ISCI did not significantly correlate with experimental (PROUST/No PROUST) condition, but did significantly differentiate between students who chose to complete as opposed to skip homework assignments altogether ($F=7.83$, MS error = 12.81, $p<.001$). This finding is attributable to the lower ISCI scores of those who did no homework.

The ISCI is uncorrelated with either exam and with indicators of academic standing. The lack of relationship may be a range restriction problem attributable to the uniformly high level of reported function found in the Yale sample.

Computer Attitude Questionnaire. Eighty-three students completed a 41-item survey on opinions and attitudes relating to computers (consisting of 11 CSE-developed items and the 30 items from the Gressard & Loyd, 1984 Scale). All but two of the questions about opinions show means very close

to the scales' centers, with generally equal numbers endorsing either direction. Questions which relate to the importance of computer technology were usually answered in the affirmative (means of 4.5 and 4.2, respectively on five-point scales.)

Attitudes were appraised using Gressard and Loyd's four-point scales, and each point on every scale was endorsed by at least one respondent. Only a quarter of the scales show means substantially above or below the scales' midpoints. These are:

- I'm no good with computers (disagree)
- Working with computers would make me very nervous (disagree)
- Challenge of solving problems does not appeal to me (disagree)
- I feel hostile and aggressive towards computers (disagree)
- I am sure I could learn a computer language (agree)
- Using a computer would be very hard for me (disagree)
- I get a sinking feeling when I think of trying to use a computer (disagree)
- I do not think I could handle a computer course (disagree)

The average total score for the Computer Attitude Survey was 131.13 out of 175, (sd 17.21). It was moderately but significantly correlated with both midterm and final exams ($r=.35$ and $.39$ respectively). The total score was not significantly different by experimental condition or homework turned in.

PROUST Questionnaire. The PROUST questionnaire was an instrument designed to focus directly on the PROUST experience and was completed only

by students in that treatment condition. The three page questionnaire covered student reactions to the effectiveness of the experience and selected affective and metacognitive factors. The category range was "very effective" to "not effective."

Overall student results were positive; they said PROUST helped them write programs faster, and they recommended PROUST's contribution to the task of debugging their programs. They felt that PROUST feedback was easy to understand. PROUST was given only average marks in identifying their programming bugs (perhaps because plans were misinferred.) Enjoyment and ease of use were given average (neutral) scores on the whole, with students endorsing all possible points on each scale. Students were also divided about the appropriateness, sufficiency and relatedness of the feedback. On two items, whether PROUST aided them in becoming more comfortable using the computer and in writing programs, no student marked "very effective." Students were divided in the programming strategy responses, in whether they preplanned their program on paper, and whether they collaborated with others (an additional source of confounding). Almost all reported that they concentrated on breaking a programming task into component parts, a significant feature of the theoretical approach of PROUST. (See Table 11.)

In written comments on the PROUST Questionnaire, students in the PROUST program felt that the program took excessively long to load. They also reported feelings of frustration and anger when the program gave an erroneous bug. The occasional newcomer to computers reported feelings of inadequacy in the face of a very complex piece of machinery; while on

Table III-11

PROUST Questionnaire

	RAIN	RAIN + BANK	NONE	TOTAL
n of cases	6	11	10	
1. Purposes presented on screen: yes/no ¹	2/4	4/7	6/4	12/15
2. Told of purposes by instructor: yes/no ¹	6/0	11/0	9/1	26/1
3A. PROUST effective for looping ²	3.50	3.45	2.70	3.18
3B. PROUST effective for debugging ²	3.00	3.00	2.30	2.74
4. Helped to write Pascal faster: yes/no ¹	4/2	5/6	9/1	18/9
5. Recommend PROUST to others: yes/no ¹	5/1	7/4	9/1	21/6
6. Change PROUST: yes/no ¹	2/4	6/5	2/8	10/17
7. Correctly find bugs on rainfall ³	3.67	3.09	3.22	3.27
Correctly find bugs on checkbook ³	2.55	3.75	3.22	3.29
8. Feedback easy to understand: yes/no ¹	4/2	7/4	6/4	17/10
Feedback appropriate: yes/no ¹	4/2	6/5	9/1	20/7
Feedback sufficient: yes/no ¹	4/2	7/4	5/5	16/11
Feedback related to lectures: yes/no ¹	4/2	10/1	7/2	21/6
9. Enjoy using PROUST ³	2.67	2.00	3.20	2.59
10. Motivate to learn Pascal ³	2.67	2.27	2.20	2.33
11A. More comfortable using computer ³	2.50	2.00	2.60	2.33
11B. More comfortable writing programs ³	2.67	2.18	2.80	2.52
12A. Easy to use the computer ³	3.00	3.45	3.10	3.22
12B. Easy to use the editor ³	3.00	3.73	3.20	3.37
12C. Easy to find information on screen ³	3.00	3.18	2.90	3.37
13. Graphics would enhance effectiveness: yes/no ¹	1/5	1/10	5/5	7/20
14. Preplan program on paper: yes/no ¹	6/0	10/1	10/0	26/1
15A. Focus on individual program statements ¹	0/6	0/11	1/9	1/26
15B. Focus on overall program: yes/no ¹	6/0	11/0	9/1	26/1
16A. Collaborate with students on line: yes/no ¹	2/4	1/10	4/6	7/20
16B. Collaborate with students off line: yes/no ¹	4/2	4/7	5/5	13/14
16C. Collaborate with instructor on line: yes/no ¹	1/5	1/10	0/10	2/25
16C. Collaborate with instructor off line: yes/no ¹	0/6	0/11	1/10	0/27

Notes:

¹Tally of yes responses / no responses²Lower values are more positive³Higher values are more positive

the other hand, "old timers" in computer technology felt that the system was not as far advanced as they would have hoped (the arrogance of youth, no doubt) and that they would have appreciated a special lecture on the underpinnings of PROUST itself. Several students stated their pleasure at having been involved in the PROUST experience and their new appreciation for the power of computer technology.

Interpretation

Yale students are a strikingly high performing group by traditional academic standing indicators. Their success on the course examinations is clearly related to their skills developed in the course incidental to PROUST and homework completion. The midterm exam was highly related to students' SAT verbal scores, and those who already possessed a wide background in mathematics courses generally scored higher than those with less background. The only treatment difference, related to homework completion, suggested that students who do not do homework lose out. Whether this loss is because of lack of practice, low confidence and motivation (as suggested by their ISCI scores) or a combination can not be determined when homework was made optional.

The lack of clear performance differences in programming was disconcerting but may have a number of explanations. At once, a preliminary explanation was range restriction and ceiling effects (Yale students are so good that no differences could be obtained). However, the relatively low average percent of attainment (55% for midterm, 53% for final) of the performance measures quickly obliterated this option. But

other explanations, obvious in hindsight, can be posited. First, the examinations as configured may be too difficult, may ask too much of students. Secondly, the compensatory effects of seeking assistance of teaching assistants, diagnosticians and prescribers, may have wiped out differences among treatment groups. Tracking the amount of time and nature of TA involvement should be considered. Third, the no-difference finding may be attributable to the dependent measure used. PROUST students report that they believe PROUST helps them to write programs. Perhaps that together with actual written programs should be used as criterion measures rather than simply identification and explanation of bugs. Learning does not always occur in the hierarchical way we might suppose, and there may be great unmeasured PROUST effects.

Finally, an interesting contrast between students' general views of themselves as learners (high), confidence with technology (high) and reaction to PROUST (slightly positive) could be thought to be attributable to the character of the course instruction (the instructor's congeniality and expertise, for instance). PROUST certainly did not damage students' opinions of their own capacity (but it would probably take a good deal to do so given the talent of this group), but neither was PROUST a recipient of positive transfer of computer technology attitudes in general.

UCLA Study

Method

The intent of the UCLA study was to replicate, using substantially similar measures, the Yale Study. To that end, arrangements were

negotiated and completed with the UCLA mathematics - science department to provide controlled experimentation using PROUST during the spring quarter, 1985, in regular beginning PASCAL programming courses. UCLA staff reviewed PROUST documentation, homework assignments, and the midterm and final examinations, and were surprisingly agreeable to incorporating these common features. While we had requested one or two sections of students, UCLA felt that students in six sections should be given the opportunity to participate in the experiment, and the CSE staff planned to collect data on approximately 150 students. Because the experiment was embedded in regular course offerings, the timing of implementation was critical.

Yale staff were requested to send necessary software files and provide assistance in transferring the systems to the UCLA UNIX environment. A series of files were sent, with some files missing.¹ Unfortunately, a critical interface was also not available. UCLA staff worked to prepare the interface and to get the PROUST system up and running on the UCLA system. A number of technical problems were experienced, with round the clock effort being applied by UCLA staff and computer consultants.

Just at the close of the time planned for PROUST implementation, the system finally worked. In the meantime, UCLA faculty had switched the experiment from required to optional, given the uncertainty of the system performance. However, the system was running two days before the homework assignments were due.

¹ According to Yale staff, "the normal problems ... in porting software that was never meant to be ported ... were in fact encountered."

During the fifth week of the ten week quarter, 150 students in a math-science course for beginning PASCAL programmers directed to non-majors were given the opportunity to use PROUST for two homework assignments. Of these 150 students, 21 logged on the system but submitted no program assignments to PROUST. Nine submitted Rain programs to PROUST, averaging three program submissions each. In addition to the general interface problems experienced, UCLA computer usage practices are open, and students can choose to interact with the system at their own convenience. Because students apparently chose to use PROUST at a heavy use time, system performance was unsatisfactory and a number of students aborted this optional requirement in frustration. Specifically, the program took as long as three hours to load and one and a half hours to complete its analysis. The average loading time was an hour and the average analysis time was 45 minutes.

During the last three weeks of the quarter, attempts were made to interview the nine students who had interacted with PROUST for approximately an hour each to ascertain their attitudes toward and experiences with PROUST. The interview topics are appended to this report. No ancilliary data were collected on these students.

Results

Because of system transfer problems, the performance examinations were not given by UCLA faculty, thus any direct comparisons with the Yale data became impossible. There was neither a control group, nor information on the effects of any problem other than the Rain problem.

Student reactions to PROUST as implemented at UCLA were mixed. Some students were frustrated by physical problems: long load time, unable to get it to run, unable to exit it, and continued bombing on the Bank problem. Two of the students disliked the program's "personality" and said it made them feel dumb. The students also criticized PROUST's feedback and suggested that it should (a) give examples, (b) provide more instruction, not just information that could be misunderstood no matter how many times it is repeated, (c) specifically tell the student what to do to correct mistakes identified, and (d) use language appropriate to students who are not familiar with computerese. Despite this criticism, students also mentioned that PROUST provided clues that could be used in future programming and pointed out things the students had forgotten. One felt it was more personal than the compiler and said that it provided a sense of "company" at 3 a.m. when no one else was there to help.

Technology Transfer

An issue that is perpetually raised is the technology transfer requirements for any innovation. These concern technical issues, such as incomplete interfaces, local adaptation requirements, and organizational habits, like the open computing mode that contributed to the intolerably long loading and analysis times. Although we attempted to secure appropriate technical information early, Yale staff were predictably busy on their own concerns. We would have wished to have a Yale staff member

supervise the system transfer, but that proved to be impossible.¹

Documentation issues, as always, are also a concern. Nonetheless, this experience should provide assistance to the designer to anticipate system transfer problems should the project reach the stage where exportability becomes a serious interest.

Conclusions from the Effectiveness Studies

PROUST had no demonstrable main effects on the aspect of programming it was designed to affect; but confounding issues raised in the description of the Yale study suggest a number of explanations. An additional fact may be that a one-time only (or twice-only) experience is not sufficiently strong instructionally to exhibit effects. Student attitudes were generally positive but not characterized by enthusiasm for PROUST effects. It may well be that the complexity of the technology is mismatched with the novice level of the students. They perhaps cannot truly appreciate it yet. Of course, raising the experience level of students would concomitantly raise the complexity of the programs they submit, undoubtedly taxing PROUST's present level of functioning.

¹ No specific resources were budgeted for Yale to provide this.

CONCLUSIONS AND RECOMMENDATIONS

Given the three phases of this study, theoretical analysis, formative review, and effectiveness study, what can be said about PROUST and what might be predicted or suggested about its future?

PROUST really does do a good deal of what is claimed in its name. It takes programs, processing a large percentage of them, and with remarkable consistency identifies bugs in students' approaches. The effects of this process do not show up on the measures thought to be most appropriate by the system designers, but suggestions for actual programming outcomes might be considered. PROUST is presently implemented to deal with two comparable problems, but does so with rather different degrees of success. The Bank problem seems to be less amenable to reliable analysis. In an effort to understand whether the problems were really analogous, CSE had experts generate, based on the Rain problem, what they thought would be a comparable problem. They designed another Bank problem, even to the topic, a verification that experts think these two problems are comparable. Why aren't they? It may be that aspects of the topic have greater effects than issues regarding common algorithms. Thus, it would be important for PROUST to focus on showing that it can generalize across a range of problems. A sophisticated approach that reliably interacts with one and a half problems raises questions (when one is in an instructional utility, rather than a research or knowledge production, mode). PROUST should avoid the pitfall of human expertise which tends to focus on learning more and more about less and less.

We have also suggested that PROUST in its present form might be timed differently, might attempt to be more interactive (verifying intention inference with the student for instance) and might be incorporated following input/output analysis by the student. In the technical expansion of PROUST, the creation of a cumulative student model that tracks effects on repeated submissions rather than as isolated events would show promise and seems to be a necessary step to the development of the tutorial component.

In the development plan for future versions of PROUST, the designers have a number of choices. Should they seek to generalize to more problems? Should they move to the implementation of a complete student model? Should they attempt to develop the tutor? These are not mutually exclusive choices and our recommendation is that the designers proceed in parallel on all three. Any first guess about the tutor will be imperfect at best, but undoubtedly the learning here will have implications for PROUST design modifications. Generalizing PROUST would be an important theoretical, practical, and marketing accomplishment. Last, the authors should be commended for their interest in student effects, and not discouraged by the findings. Better (and or different) criterion measures may be created, and there is relatively no risk in using them.

If future PROUST effectiveness studies were conducted, they might need to move toward laboratory experiments, where confounding (related to student choice of completing assignments and use of teaching assistants) could be controlled. PROUST really should be tested in another (non-Yale)

environment at some point. This requires confronting the problems of technology transfer and the benefit of finding how PROUST works on a less spectacular set of students.

With additional development, PROUST shows some promise, despite its problem, for future Army needs provided it were accompanied with standard direct instruction. Effective implementation, however, will require clear specification of how PROUST is to be integrated in the classroom.

The PROUST staff took a risk in participating in this evaluation. It is uncharacteristic for anyone to volunteer for evaluation in any venue, and even more rare among the AI community, whose present commitment to research goals is strong, whose sense of the "rightness" of views and approaches of those inside the community contrast with their skepticism and impatience with those outside the community. If AI and ICAI deliver on their promises and the excitement they have generated, no need for evaluation studies of this sort will exist. However, if progress in AI is delayed, the support of those inside the community may no longer be sufficient. Then convincing demonstrations of system performance and effects may be needed. The Yale staff has taken an important, and probably, unpopular step. They should be rewarded for their commitment to effectiveness of their enterprise.

REFERENCES

- Ernst, G. & Newell, A. (1969). GPS: A case study in generality and problem solving. New York: Academic Press.
- Gressard, C.P. & Loyd, B.H. (1984). An investigation of the effects of math anxiety and sex on computer attitudes. Paper presented at the annual meeting of AERA, New Orleans, LA.
- Hiller, J.H. (1974). Learning from prose text: effects of readability level, inserted question difficulty and individual differences. Journal of Educational Psychology, 66, 202-211.
- Hovland, C.I., Lumsdaine, A.A., & Sheffield, F.D. (1949). Experiments in mass communication. Princeton, N.J.: Princeton University Press.
- Johnson, W.L. (1985, May). Intention - based diagnosis of errors in Novice Programs. New Haven: Yale. U/CSD/technical report, #395.
- Johnson W.L. & Soloway, E. (1983, August) PROUST: Knowledge-based program understanding. New Haven: Yale U/CSD/PP #285.
- Johnson, W.L. & Soloway, E. (1985, April). PROUST: an automatic debugger for Pascal programs. Byte, 179-190.
- Newell, A., Shaw, J.C. & Simon, H.A. (1957). Preliminary description of general problem solving program-I (GPS-I), Report CIP Working Paper 7, Pittsburgh, PA: Carnegie Institute of Technology.
- Norman, D. (1976). Memory and attention. New York: John Wiley and Sons.
- Schank, R.G. & Abelson, R.P. (1977). Scripts, plans, goals and understanding: an inquiry in human knowledge structures. New Jersey, L. Erlbaum.
- Sleeman, D. & Brown, J.S. (Eds.) (1982). Intelligent tutoring systems. New York: Academic Press.

IV. SHAPING THE WIND: FORMATIVE EVALUATION OF
INTELLIGENT COMPUTER ASSISTED INSTRUCTION (ICA

Intelligent Computer-Assisted
Instruction (ICAI) Study

Evaluation processes are potentially valuable productive mechanisms for the improvement of educational systems and products. And there is hard evidence of the utility of evaluation in actually improving technology based products and efforts in instructional development. However, evaluation is known as well as to contain a strong negative potential. Evaluation can identify weaknesses in such a way as to inhibit exploratory behavior and risk taking on the part of researchers and developers. Playing it safe may be seen to be the winning strategy. Evidence of evaluation utilization studies suggests that when the focus of the evaluation is classification or accountability (summative vs. bad; useful vs. wasteful), the openness of R & D project personnel to evaluation processes is inhibited. Formative evaluation, on the other hand, is evaluation whose specific function is to identify strengths and weaknesses for the purpose of improving the product or system under development (Baker, 1974; Baker and Alkin, 1973; S.M. Markle, 1967; Baker and Soloutos, 1974). The trick, of course, is in determining what should be studied, in what context the evaluation should take place, when evaluation processes are most useful, and useful hypotheses about what improvement options logically and feasibly may be implemented. In addition, evaluators need to be aware that the identification of weaknesses (no matter how benign the intentions of the evaluation may be) creates a documentary trail that might be misused by project managers or funding agency monitors.

These issues take on special dimensions when the evaluation addresses the effectiveness of new technology. All technology development of necessity focuses on the initial problem of system operation: can the envisioned delivery system work at all, as opposed to the refinement of

what the system's merits may be or what effects might be planned or imagined. The boundaries between technology development and science become especially blurred. The creation of technology may be a pleasant side-effect for the creator, whose perception of his/her main task may be knowledge production. Intellectual exploration is at a premium for new technology development; thus, evaluation processes can be seen to inhibit or be irrelevant to invention.

Recent writing in the field of evaluation planning has emphasized a stakeholder perspective in evaluation implementation. Simply put, this means that interested parties must have an opportunity to understand and to shape the nature of the evaluation questions and methods so that they will be more invested in the process and more apt to use any results generated (Byrk, 1983).

With this discussion as context, a preliminary model of evaluation was designed to be adapted especially to the problem of new technologies. This paper will detail the features of this model and illustrate their use in the problem of evaluating intelligent computer assisted instruction, a particularly difficult area characterized by weak boundary conditions. From this illustration, specific recommendations with regard to design of ICAI systems will also be made.

Features of a Model for the Formative Evaluation of New Technologies

1. Documentary Data Base. Evaluative information should provide an enhanced documentary base for the processes of new technology development. A characteristic of new technology is lack of documentation describing the process leading to the development of the system or product. The purpose

of a strong documentary base is to provide the trace of developmental processes so that the field benefits from historical lessons. Aggregating across a series of case histories of projects can allow an inference about productive strategies to be made. In addition, a good documentary base can inform about dead-ends in substance as well as in developmental processes. Since most R&D reporting is based upon positive findings, it is difficult to avoid useless, but unreported paths.

All of us are familiar with documents of development which retrospectively rationalize and make "neat" processes that are chaotic, or at best, hard to track. But in-process documentation is hard to find. This lack of documentation exists for a variety of reasons. First, the process of early design of technology is complex, iterative, and non-linear. Furthermore, the conscious awareness required of designers to document their own processes, while at the same time working on problems of interest, presents an almost insurmountable attention burden, even if there were a predisposition on the part of the research and development personnel to do so; solving the problems at hand compels their attention. Contributing to an abstraction such as "R & D processes" attracts less compelling energy, despite the intellectual apprehension that the field overall can be improved by "lessons learned". Beyond this problem deriving from lack of incentive, a strong disincentive is the competitive advantage of unshared proprietary knowledge, well known in the private sector, but of potentially increasing import in a public R & D environment characterized by competitive procurement policies.

In an attempt to meet this overall goal in instructional technology, some "case histories" were prepared 20 years ago (see D. Markle, 1967) and

an historian was even on the payroll of another large R & D facility. But these persons can be viewed as pestering and diverting, just as media reporters, trying to get the idea of what's going on without true understanding of the processes involved. In new technology development, this problem would ordinarily be exacerbated.

Fully participating formative evaluators provide another model, however, if they are linked early on in the development process and if the R & D management and staff understand the intent is to assist as well as to document process. How this can operate will be detailed in a later section of this paper.

2. State-of-the-Art Evaluation Methodology. Evaluative information should use state-of-the-art evaluation methodology, i.e., both quantitative and qualitative approaches. One of the reasons evaluation processes have been received with healthy skepticism is that they appear to be so content-free, on the one hand, and methodology-driven, on the other. The history of evaluation, as in any new mode of inquiry, is replete with "new" models that propound a particular methodological view of the world. A good deal of the discredit done to evaluation has occurred with the support and consent of its most famous practitioners, who advocated one or another highly quantitative design and analysis method as the preferred mode for solving all evaluation problems (see Baker, 1983 for a list).

Any evaluation design and its analytical methods should be driven by what information is required by whom by when, by the credibility needed by the information analysts to do their job, and most importantly by the nature of the project or activity under review (Cronbach, 1980). Such precepts would suggest an eclectic approach, mixing journalistic,

documentary and effectiveness information as appropriate.

3. Policy Feedback. The evaluative information should provide policy feedback to the supporting agencies. This feature assumes that the funding source is the contracting agency and that the formative evaluation is not a totally in-house activity. What kinds of policy feedback are appropriate? That depends in part on the nature of the policy. Clearly, issues of project staffing derive from the kind of evaluation to be conducted and the kind of technical expertise required. However, at a minimum, the formative evaluators, whatever their specialties, should attend to the fidelity of the process by the project to the project's stated goals and procedures and to the kinds of contractual, monitoring, and other oversight arrangements that might be useful in the future. Furthermore, the evaluation report can consider specifically the features or tasks that might be included in the specification of future activities of the sort evaluated.

The tension of providing such information in a way that does not undo either the project activities under study or the receptivity of future projects to evaluation must be confronted. A fine line needs to be walked, keeping track of both the professional ethics applicable to contracting agency relationships (telling the truth) and to maintaining positive connections to the target R & D communities.

4. Development of Alternative Development Strategies. The information must provide timely and useful alternatives for the formative evaluation of the project(s) under study. This platitude takes serious effort to implement. It depends in no small measure in being informed accurately and intimately with the state of development of the project; and in the evaluation staff's sensitivity to the form as well as the substance

of findings that might be useful to the project staff. This requirement also depends strongly on the level or stage of development of the technology activity. Early on, certain suggestions can be made and have potentially large effects. However, later on, the evidentiary base of such recommendations is likely to be weak. Although good evidence of project benefits and weaknesses can be more fully drawn as a project matures, modification of the technology may be considerably less likely, may cost more, and so on, as time passes and investments are made.

5. Generality of Findings. The evaluation design should provide provide generalizations that can be tested in the development of other similar technology-based efforts. Findings and recommendations must be cast in a way to make sense for other related projects. To be useful, particularly in the science policy arena, these recommendations must address concerns of larger audiences: other funders, other project designers, and the R & D community at large. This means that recommendations must range across technical, management, and effectiveness dimensions, and very well may break set with some of the project personnel's own point of view regarding technology policy. Again, a tension between meeting the precept and maintaining friends in the community is obvious.

The Formative Evaluation of ICAI

Background

The formative evaluation of ICAI projects was initiated with the intent to build instructional psychology into the design of ICAI efforts. This decision was apparently based upon the notion that ICAI had developed sufficient strength of underlying technology, tools and science so that

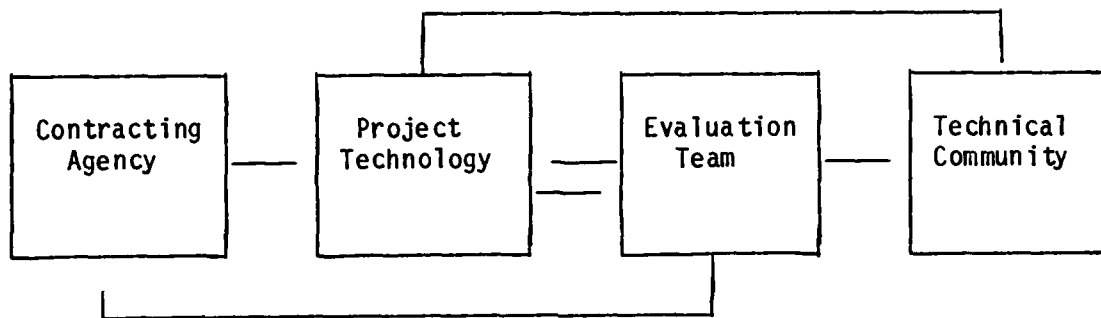


Figure IV-1
Model of relationships for the Formative Evaluation of New Technology

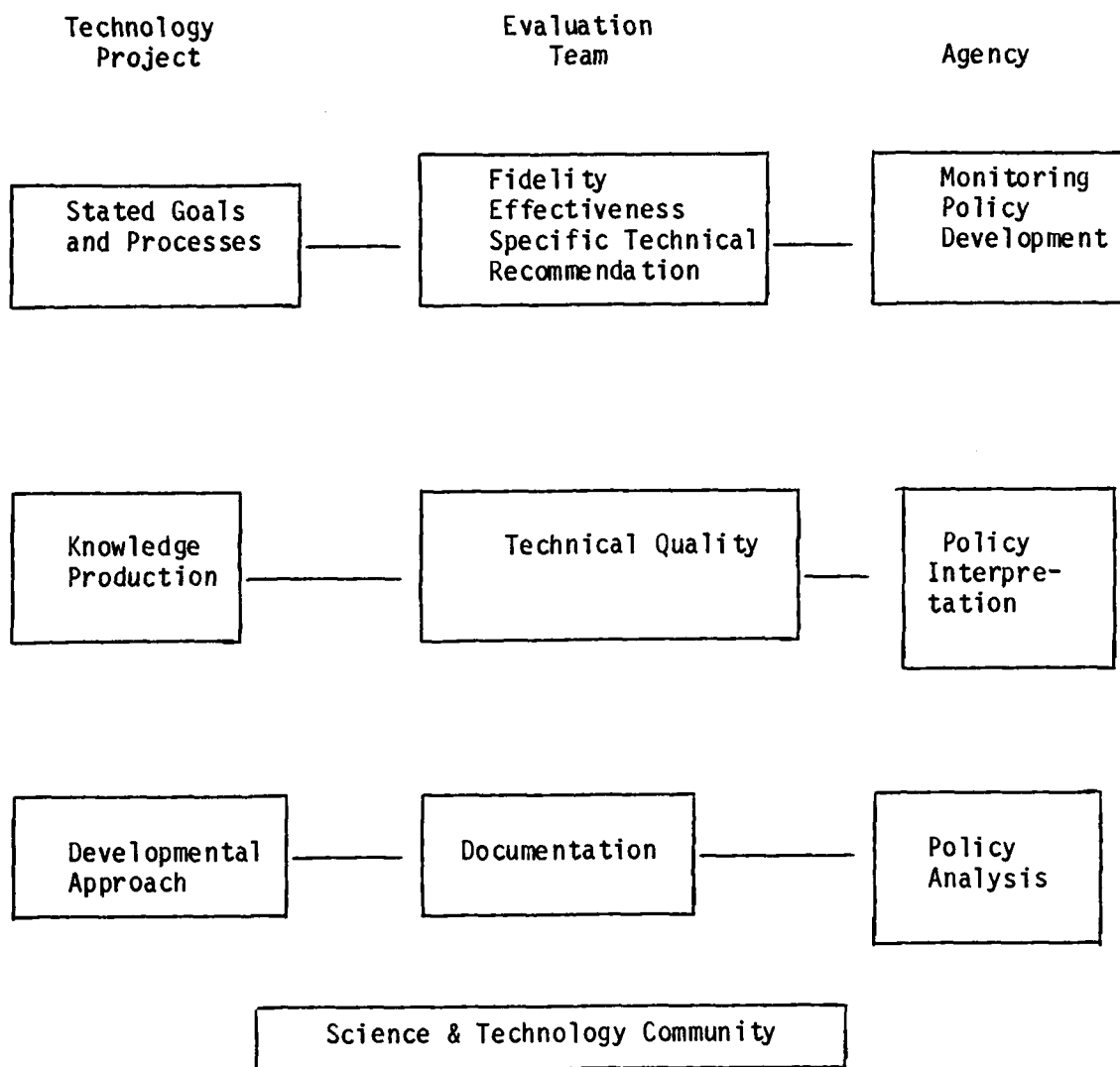


Figure IV-2
Evaluation Model Information Flow

attention could be turned to improving the effectiveness of the project(s) with research-based principles drawn from instruction. This approach was in no way intended to distract project staff from their various preoccupations with cognitive science, but rather to broaden, where appropriate, their knowledge and attention to instruction and performance assessment approaches. For instance, would the general fixation on guided discovery and student initiated instructional demands be explored? If so, what trade-offs in efficiency and effectiveness with exploratory behavior might result?

A first task, then was to decide on an overall evaluation plan. We decided that we wanted to look at multiple efforts. However, to avoid the specter of an implicit comparison, (which project was best or worse?) we intended to select projects at different stages of development. We wanted a project that was just starting out, a project that was well under development, and a project that was in some state of maturity.

Project Selection

The process of project selection itself deserves a paper, but briefly, the criteria intended and the criteria enacted differed dramatically. Like the location in real estate, cooperation was the overriding standard for selection. We also were interested in a distribution of content for ICAI, different target student groups, and to some extent different ideology or approach. In the process of exploring projects for participation, we were:

- 1) asked why should any one agree to such an evaluation?
- 2) told that ICAI, although the announced target of the particular project, was not at all of interest, it was merely a mechanism to secure funding;
- 3) and that evaluating a project might be a good marketing device, thus

participation was solicited (and rejected.) The actual sites for ICAI efforts chosen for evaluation in this project include:

1. A project at a profit-making firm
2. A project at a university
3. One of the "tried and true" ICAI efforts: WEST

Project one has a target population of maintenance technicians, and is designed to assist in learning maintenance on complex xerographic equipment. Project two has a target student group of freshman liberal arts students learning introductory PASCAL programming skills. Project three is designed for upper elementary students, and is intended to teach numerical concepts and some "strategic" level learning.

Project One

The ICAI maintenance task was at an incipient stage during the start of the evaluation project. Project staff met both with R&D personnel and with those who, at another site, will be responsible for integrating ICAI in ongoing training environments. The incipience of the project was somewhat protracted. At best, it appeared possible to solicit information via structured interview, to track documentation, and to observe, from a relative distance, the ICAI design process. An essential goal of the design team was science. The training staff had shorter-term concern and were interested in the potential for the formative evaluation to create evaluation designs and criterion measures that would assist them in the evaluation of the training in this ICAI implementation and, by extension, to future support of other ICAI projects.

One difficulty in meeting this goal was the approach to development employed, a common enough one, rapid prototyping. In the words of one

expert, that "... is formative evaluation at its best: make, try it out, and fit it." On the other hand, how one knows the "it" works, other than "runs" is problematic. A bottom up process may foster the acceptance of whatever effects occur, rather than encouraging the design and revision of the ICAI to meet explicit performance specifications. Performance specifications here means those outcomes, learning, and proficiencies demonstrated by target students rather than the system.

The decision on this project interacted with the fixed time limit for the evaluation (planned for one year.) Artful scheduling permitted a brief extension at no cost, but we did not anticipate having a ICAI implementation to test, make suggestions about, or even applaud before the end of the contract. Documentation and interviews were the fall-back strategy.

Project Two

Our hopes for project two were high. The ICAI researcher met and conversed openly about the potential for the evaluation to assist in his goals for the project. We agreed on a design that would permit 1) the specifications of dependent measures; 2) a test of the generalizability of the ICAI; 3) a set of replications, one in the university environment in which it was developed and another at UCLA, to test the exportability of the system.

This project proceeded in amiable good cheer. But problems existed as well. First, the ICAI part of the project, a tutor, was still on the drawing board, although the researcher was open to consider how such a tutor might best be built. Nonetheless, what was available for evaluation was the analyzer of the PASCAL program rather than the tutor. That

limitation may change in sufficient time for the evaluation team to make an assessment of the tutor function. Second, the set of specifications for criterion tasks stimulated a revision of the performance measure (on student programming ability) by the researchers, rather than the wholesale adoption of our recommendations. Nonetheless, progress is possible here, too. Last, the off-site replication was replete with the inevitable surprise. UCLA UNIX environment and time-sharing procedures created system transfer problems. Thus, the 150 student subject pool shrank to less than 10% of that number because of loading times that ran from one half hour to three hours at peak use. (Students were not particularly interested in doing their homework at 3 a.m. at the computing facility.) However, comparative cross-site data were generated, and recommendations for strengthened criterion measures and tutor characteristics will be made. As a side effect, a proposal for the design, analysis, and interpretation for instructional improvement of a relational database on student performance data, protocols, and individual difference measures was developed and is presently under review.

Project Three

We turned to WEST because at one level it is a known commodity. We developed criterion measures assessing both content (math) and strategic (game playing skills). We believed WEST permitted us to "play" at alternative instructional strategies and have planned variations where children will be experimentally divided into groups where they will engage with WEST under various experimental variations. Disabling the tutor is one focus: What does the AI add? Alternatives such as a job performance aid to model the information provided in the "coach" will be

tested. We have developed dependent measures that assess math performance in a broad way, and will also look at how well or what pieces of strategic thinking WEST produces. It is our intent to look at individual difference predictors of outcome performance. This project too was not without its problems. The WEST software was developed over time, patched and not well documented. Good assistance has been secured from the software developers to assist our staff's interest in modification to introduce experimental variations. But minor annoyances persist. Delivery dates for hardware were not met; the schedules for running 75 children as subjects, fixed during the start of the school year, needed constant modification, due to hardware delays. Thus, findings from this work are not available.

However, in WEST we had an opportunity to explore the instructional power of our ideas against a set of clear external performance criteria. We have as well the mixed blessing that WEST belongs to everyone, so that the threat of psychic investment is not a problem. Last, since no agency is looking at WEST as an illustration of its own research policy, agency interest is less high, but still of sufficient level to make our efforts useful.

Other Tasks

One goal of the evaluation project was to use our findings from empirical work and documentation to permit the design of specifications for an AI instructional editor. This work is now less based on the linear flow of information from our project than the overall integration of ICAI experience from technical experts and the knowledge the evaluation team has about instruction and measurement options.

Recommendations

Our recommendations are of necessity highly tentative. It does appear that a focus on good dependent measurement is an area of real weakness in ICAI activity. We recommend that future activity be required to use measures that have content and psychometric validity.

In the instructional realm, it is clear that AI researchers may need to choose, or at least agree to, a more direct instructional approach for some of the goals of ICAI systems. The trade-off between developing the earth shattering perfect student model(s) vs. focussing the system on producing (not just tracking) learning must be directly considered.

In reviewing funding agency roles, three recommendations are imperative. One, the formative evaluation period for contracts must be undertaken with the knowledge that everything will take more time than it should, and that evaluation can not be done effectively when the "its" to be built reside in the heads of R & D personnel.

Since cooperation and credibility turn out to be so important, second, agencies must be much more aggressive in insuring understanding and linkage between research project and evaluation personnel. Goodwill between groups is simply not sufficient to sustain the effort.

Third, agencies, must design their procurements such that they meet both the needs of researchers (to expand science) and the needs of project management with concern for effectiveness and efficiency of the ICAI project.

References

- Baker, E.L. Formative evaluation in instructional development (with M.C. Alkin). AV Communication Review, 1973, 21(4).
- Baker, E.L. Formative evaluation of instruction. In J. Popham (Ed.), Evaluation in education, McCutchan, 1974.
- Baker, E.L. Formative evaluation of instruction. (With W.A. Soloutos) Los Angeles: UCLA Center for the Study of Evaluation, 1974.
- Baker, E.L. Evaluating educational quality: A rational design. Invited paper, Educational Policy and Management, University of Oregon, October, 1983.
- Byrk, A. (Ed.) Stakeholder-based evaluation. New Directions for Program Evaluation. Vol. 17. San Francisco: Jossey-Bass, 1983.
- Cronbach, L.S., et al. Toward reform of program evaluation. San Francisco: Jossey Bass, 1980.
- Markle, D.G. An exercise in the application of empirical methods to instructional systems design. Final report: The development of the Bell System first aid and personal safety course, American Institutes for Research, Palo Alto, CA. New York: American Telephone and Telegraph Co., April, 1967.
- Markle, S.M. Empirical testing of programs. In P.C. Lange (Ed.), Programmed instruction. The Sixty-sixth Yearbook of the National Society for the Study of Education, Part II. Chicago: NSSE, 1967.

V. EVALUATION OF WEST

Intelligent Computer-Assisted Instruction (ICAI) Study

CHAPTER SUMMARY

This report presents the description and evaluation of WEST, an ICAI game designed by Burton and Brown to teach basic mathematics skills. There are three major sections of this document: a review of related research, a theoretical and formative review of the game, and a report of two effectiveness studies conducted with WEST.

The purpose of this project was to evaluate WEST in terms of the learning and instruction processes incorporated in the program. In addition, an experimental contrast with various instructional options for WEST was designed, and two studies were conducted to measure the effectiveness of the program with elementary school students.

The theoretical orientation of WEST is based on the premise that students can learn from their mistakes, or "bugs". The intelligent aspect of the program addresses the identification, diagnosis, and treatment of the "bugs". The AI components of WEST - knowledge representation, student model, and tutoring strategies - are described and discussed. Critical instructional variables are also discussed, including learning objectives, skill specification, documentation and directions, text displays, learner control, tutoring strategies, and models of student behavior.

Effectiveness studies showed few significant findings related to learning outcomes, but observational data and computer transcripts provided interesting findings in relation to students' attitudes and learning needs in the WEST environment. Implications for future directions are presented in the concluding remarks.

INTRODUCTION

This report describes the formative evaluation of WEST, an ICAI program to teach arithmetic skills to elementary age students. Students are presented with a game board and the object of their attention is to win, or reach the end of the game before their opponent, the computer. Students have various strategy options they may employ to win, but a necessary condition is that they solve a number sentence generated by random spinners. The level of difficulty of the game can be increased. What makes WEST especially interesting is that the computer tracks student moves and provides them with coaching regarding better moves they might make. This interactive, adaptive capability is what gives the intelligence to WEST. How the game works and what its cognitive demands are will be more fully described in a later section.

Evaluation Focus

This report is divided into three major sections: a review of the literature of related learning research, a theoretical and formative review of the game which describes the principles underlying game construction and interaction, and a report of effectiveness studies describing the outcomes achieved by students playing the game. It was the initial intent of this project to study the learning processes incorporated in WEST, and to design an experimental contrast where strengthening instructional options for WEST were implemented. The effectiveness studies also attempted to present a broad base of outcome measures, assessing the mathematical learning outcomes of the game, the strategy learning outcomes of the game, process

data on how and how skillfully students played the game, and the interaction of these with individual difference characteristics of students. Because of unforeseen but previously documented (in our progress reports) delays of the equipment necessary to conduct WEST experiments, the schedule for the studies was greatly compressed. Had equipment arrived earlier, we had hoped to conduct a sequence of revise-test implementations, successively trying various combinations of treatment variations to attempt to improve student performance values.

Because WEST is a "mature" game in the lifespan of ICAI, certain benefits and problems accrued. As a benefit, there was no development schedule upon which we were dependent. There was also no real proprietary or protective sense of the WEST designers or concern with our activity. On the other hand, we did benefit from assistance from one of the original designers (Burton) and from the designer of the CAI precursor to WEST, *How the West Was Won* (Bonnie Anderson).

The specific evaluation questions guiding this study are presented below:

1. What is the underlying theoretical orientation of WEST? To what extent does the program serve as a model of ICAI development?
2. What instructional strategies and principles are incorporated into the program? To what extent does the program exhibit instructional content and features potentially useful to future Army applications?
3. What are the learning outcomes for students? To what extent do learners achieve program goals? Do students with different background characteristics profit differentially from exposure to WEST? To what

extent does the program create unanticipated outcomes, either positive or negative?

The specifications for the conduct of the study are presented in Table 1. Here we describe the basis of data collection for each of the questions provided.

The next section presents a literature review related to WEST concepts, followed by the other named sections above.

Table V-1
Instrumentation and Data Collection Strategy

Evaluation Question	Dimensions of Inquiry	Measurement Method	Data Source
1. What is the underlying theoretical orientation of WEST? To what extent does the program serve as a model of ICAI development?	<ul style="list-style-type: none"> theoretical view of learning and instruction ICAI components 	content review interviews	primary documents learning research literature project developers AI experts
2. What instructional strategies and principles are incorporated into the program? To what extent does the program exhibit instructional content and features potentially useful to future Army applications?	<ul style="list-style-type: none"> subject matter content instructional strategies and principles student reactions Army needs 	program review transcript review observation interviews	subject matter experts (instruction and AI) elementary school students
3. What are the learning outcomes for students? To what extent do learners achieve project goals? Do students with different background characteristics profit differentially from exposure to the project? To what extent does the program create unanticipated outcomes, either positive or negative?	<ul style="list-style-type: none"> math skills and game strategy background characteristics (math achievement, computer, math, game attitudes) game behavior reactions to WEST 	paper and pencil tests questionnaire standardized tests transcripts of game interview	elementary school students

LITERATURE REVIEW

WEST is an ICAI game that requires players to construct number expressions to make moves along a game board. The object of the game is to be the first to reach the end. WEST has an "intelligent" component called a computer coach. The coach was developed by Richard Burton and John Seely Brown for a game called "How the West Was Won". This instructional game is part of the PLATO computer-based learning system at the University of Illinois (Dugdale & Kibbey, 1977; Seiler & Weaver, 1976). The philosophy behind the development of the WEST coaching system assumes that the student constructs new knowledge from prior knowledge. This construction of knowledge is assumed to contain errors, or "bugs". The WEST coach was designed to help students learn from their errors by guiding them to see what caused their bugs. The approach is termed "guided discovery learning" (Burton & Brown, 1982).*

The Artificial Intelligence (AI) technique incorporated into WEST is differential modeling, or overlay approach, in which a model of student performance is built by the system in comparison to "expert" performance. In addition, specific tutoring principles were adhered to when designing the coach's intervention. These involved issues (what should be said) examples (how it should be said), and timing (when it should be said). Through its approach the WEST program addressed the problem that "if left

* Burton and Brown's article was first published as a special issue of the International Journal of Man-Machine Studies, Vol. 11, January, 1979.

alone, students fail to see interesting structure in the environment" (Brown, Burton and Clancey, 1984).

The question of structure in the WEST environment was originally explored when the game was still part of the PLATO system (Resnick, 1975). Resnick contended that CAI programs paid little or no attention to how a learner structures the learning environment. Without an adequate model of the student, good teaching is "impossible" (p.3). In the Resnick study, 38 students (8 to 11 years of age) were observed while playing "How the West Was Won". Each student played against the computer and patterns of play were recorded. The patterns were analyzed in terms of strategy, optimality and change in behavior. Resnick constructed computational models of students which described the various ways students imposed structure on the learning situation. The models were then used to confirm categories of player behavior -- i.e., classifications that described students' representations of structures of the learning environment. The results of the study showed that most players established an early pattern of play in which they tended to stay. They continued rehearsing moves they understood and did not experiment with new forms of arithmetic expressions. Further, they ignored the wide variety of moves and expressions the computer used. Resnick concluded that student models were necessary to begin creating responsive learning environments.

Understanding student thinking has been addressed in more recent research on students' mathematical strategies and errors (Brown and Burton, 1978; Carpenter and Moser, 1982). These studies support the idea that the student is an active participant in constructing meaning, but often the

understanding is flawed. If flaws in thinking are not recognized in the instructional situation, then learning may not be optimal.

A major intent of the WEST environment is to have the student actively engaged in learning. Specifying levels of student participation must include the degree of interaction between the learner and the program. The opportunity for WEST players to ask for hints, together with the interventions of the coach, provide various types and levels of available explanations of math skills and game strategies. When and how often players receive these explanations may affect performance and learning outcomes. Research on the role of explanations in learning situations contains consideration for the design, implementation and evaluation of interactive learning materials. Webb (1984) reported two findings related to these issues: receiving explanations is sometimes positively related to achievement; and, receiving no explanation when one is needed is consistently negatively related to achievement. The study discussed interaction in a computer setting and noted that students seemed to learn from what others did as well as from what was said. This has implications for WEST-like situations in which the computer models desired behavior.

Burton and Brown designed the WEST coach to tutor students on their skill and strategy flaws or "bugs". When a student plays WEST, the program tracks the types of number sentences and strategies used. When a student persists in using a less than optimal move or expression, the coach interrupts and attempts to guide the student toward better use of skill and strategy. A model of student performance is compiled to describe variety of expressions used and optimality of strategies employed.

There has been minimal research published on the WEST program. An experiment was conducted with 18 student teachers to test the coach's ability to identify learning bugs. Results of a questionnaire indicated most subjects felt the coach understood their weaknesses (Burton & Brown, 1982).

A controlled experiment was conducted with elementary school children in which coached and uncoached versions of WEST were compared (Goldstein, 1979). Two major findings highlighted in this study were: the coached group displayed a greater variety of mathematical expressions used; and the coached group expressed greater enjoyment in playing the game. These reports lead to the question -- what are the intended learning outcomes of WEST? Originally, the PLATO version was used to review and practice computational skills in a fun environment. However, in addition to arithmetic skills, there are strategic skills to be learned. The ICAI version of WEST addresses both arithmetic and strategy, but available literature fails to adequately specify variables and skills in either category.

Game strategy is crucial in WEST. Research on game-playing skill acquisition is sparse but a recent review of cognitive aspects of games concluded: people play games to maximize goal attainment; people are sensitive to opponent strategy; people are frequently not capable of identifying optimizing strategies; and, people play games based on their internal representation of the task environment (Laughery, 1984). The implications of these conclusions for the WEST player include specifying goals, defining optimal strategies, and building a meaningful representation of the task.

Research on cognitive processing can be approached from the perspective that cognitive strategies can "enhance" human information-processing. For example, Ann Brown and her colleagues devised instructional routines to help students construct meaning from texts (Brown et al, 1981). Through the use of rehearsal, categorization and elaboration techniques junior college students were trained in using basic rules for summarizing information. Remedial and average students received various levels of training, from self-management to explicit instruction. The general pattern of results indicated remedial students benefited from all forms of training in structuring meaning. However, for the average students to be brought up to the level of four-year college students, the most explicit instruction was needed. This research emphasized the importance of explicit intervention for the improvement and generalization of strategies for summarizing and recalling important information from texts. Perhaps the success of the students was related not only to the strategies they learned but also to their building structure for the learning situation. Perhaps lack of an explicit intervention for unsuccessful students prevented their building a meaningful framework for using the strategies.

Building a "conceptual framework" is the concern of recent research involving aspects of schema theory (Hewson and Posner, 1984). In the study, students using the same instructional materials were interviewed about the effectiveness of the materials in teaching important physics concepts. Not surprisingly, different opinions emerged. Analysis of the student responses showed that individuals exhibited different needs relating to organization, perception, assistance, association and

conceptualizing. Because these students were novices, they were further analyzed in comparison to an "expert" in understanding these concepts. A prime difference found between the two levels was the expert's representation of central conceptions. In addition, expert problem-solvers began with qualitative representations followed by quantitative statements.

The researchers concluded by stressing the need to teach novices fundamental conceptions or "synoptic views" of subject matter so they form a framework on which to build detail and sophistication. Like the first study cited, this project highlighted the need to help students structure their learning and understanding. This need does not exist only at higher education levels. Perhaps if younger students are "explicitly instructed" in building "frameworks of conceptions" in novice learning situations they will approach more meaningful and expert levels of understanding and performance. These ideas were incorporated in the effectiveness studies reported here.

THEORETICAL ANALYSIS AND FORMATIVE REVIEW

Evaluation Questions

1. What is the underlying theoretical orientation of WEST? To what extent does the program serve as a model for ICAI?
2. What instructional strategies and principles are incorporated into the program? To what extent does the program exhibit instructional content and features potentially useful to future Army applications?

This section analyzes the theoretical orientation of WEST (the ICAI version developed by Burton and Brown) as a whole as well as each of its components: student model, knowledge representation, and instructional strategies. A formative review is incorporated in the analysis of each component. This review was based on a wide set of variables which have been demonstrated to be necessary for effective instruction. A comprehensive formative evaluation tool was compiled from several sources (Bass & Dill, 1984; Cohen, 1982; Markle, 1983; Merrill, 1983; O'Neil, 1979; Reigeluth, 1983; Walker & Hess, 1984) and is included in the Appendix to this report. Because the evaluation tool is so lengthy and comprehensive, those variables which are most salient to an evaluation of WEST are discussed below.

WEST, one of the first computer coaches, provides an interesting and important context for examining the potential of an intelligent tutoring system since it is an important prototype in the field and is available in the public domain.

WEST is a prime example of a computer-based learning environment in which the student is involved in an activity (in WEST's case, a computer game) during which the instructional program observes his/her behavior and occasionally offers criticisms or suggestions for improvement. This type of game provides students the opportunity to form strategies and knowledge structures that have general usefulness in other domains as well. In WEST's case, these are broadly described math skills and general game playing strategies.

Detailed learner outcomes, specific applications of the program and entry level skills were not identified by the developers. Objectives for the learner, other than winning the game, are not stated. WEST's major emphasis (e.g., recall, application, transfer, critical thinking) is not explicitly stated and is not documented in any support materials. In addition, various levels of learning (e.g., responses, rules, cognitive strategies) are not clearly presented. This lack of essential instructional considerations hampered a comprehensive evaluation of the program's intents.

WEST follows the general paradigm for constructing tutorial systems called "Issues and Examples" (Burton & Brown, 1982). WEST content is the collection of "Issues" (concepts and skills) selected for the program. Instructional presentations of Issues are provided using "Examples" (explanations and illustrations). Criteria for selection and presentation of the program's content are not defined. For example, range and sequence of material do not support specific goals. Also, appropriateness of content and learner control are not addressed by the developers. Instructional issues will be further highlighted in the following sections.

Theoretical Orientation

The pedagogical theory underlying WEST is termed "guided discovery learning." This approach shares with discovery learning the assumption that the student builds understanding of a situation or problem from prior knowledge through an active, problem-solving process. Thus, the student should be free to interact in a new situation, making both correct and incorrect decisions, and to observe their results in order to add new knowledge to his/her domain.

Research on discovery learning has shown mixed results (e.g., Wittrock, 1963; Worthen, 1968). For example, Shymansky, Kyle and Alport (1982) have reviewed numerous studies indicating that this method can be particularly effective with science, and Guthrie (1967) found that students taught by the discovery method were able to transfer problem solving skills to new problem situations. But Montague, Ellis, and Wulfeck (1981) pointed out that discovery learning research has differed greatly in content, and few researchers have analyzed the content or problem solving processes actually used by subjects. Anderson and Faust (1973) suggested that consideration should be given to the difficulty of the principles to be learned and the guidance provided to subjects to make discoveries. Inconsistencies and lack of specification of variables have made it difficult to derive direct prescriptive recommendations to guide the design and development of instruction.

WEST's developers attempted to specify a discovery learning environment that provided guidance to students. They felt that when left alone, students tend to fail to see interesting structure in the

environment (Brown, Burton & Clancey, 1984). If the student does not have enough information to learn from his/her mistake, a tutor or coach could provide that information to facilitate learning. In their language, if the student has enough information to determine what caused the mistake, the bug is referred to as "constructive", and s/he can correct it. If, on the other hand, the student does not have sufficient information regarding the cause of the error, (a "non-constructive bug"), s/he may not be able to correct the behavior. A key assumption of guided discovery learning is that constructive bugs are a significant aspect of a learning environment. Hence, the role of the tutor or coach in an informal environment such as a game is to give the student some information in confusing situations to transform non-constructive bugs into constructive ones. It is essential to note here, however, that giving information does not necessarily constitute an instructional presentation. Constructing meaning is dependent upon various cognitive processing factors not addressed by WEST.

Preinstructional strategies are lacking and text displays are not formatted to enhance concept learning and retention. For example, students are not guided to attend to critical features nor are they given an opportunity to actively construct questions and responses. So, receiving information is not enough to correct errors.

In the present study, long "non-constructive" error routines were frequent. For example, the student whose partial transcript is presented in Figure 1 used the same number sentence pattern ($A+B \times C$) for seven turns in succession. The student was tutored on subtraction and then had an interaction with the computer as shown. With insufficient information

Figure V-1

Message Displayed on Screen	"Non-Constructive" Error Routine Annotations
You must have made a mistake. (1+1)-3 is NOT 0 Would you like to change your expression? --> Yes	Student is attempting a number sentence using subtraction after having been tutored on it earlier. Makes error.
You must have made a mistake. (1+1)-3 is NOT 0 Would you like to change your expression? --> Yes	Student repeats error. Gets no instruction or corrective feedback.
You must have made a mistake. (1+1)-3 is NOT 0 Would you like to change your expression? --> Yes	Student repeats error. Gets no help.
You must have made a mistake. 1+1*3 is Not 49 Would you like to change your expression? --> Yes	Student changes expression but makes error. Gets no help.
1+1*3=4	Subject reverts to original pattern (without subtraction) and succeeds - thus original pattern is reinforced.

about the errors, the student returned to the original pattern and continued it throughout the entire session.

A more frustrating situation arose when insufficient information was accompanied by loss of one's turn. Note the end of the exchange in Figure 2.

Each of the above situations took five to six minutes of the student's game playing time. Often, students had several of these error routines during their sessions. Given the relative lack of learning which seemed to take place, one must seriously question the rationale for this approach.

Certain cognitions and affective factors were indirectly addressed in WEST by the placement of a constraint on the coach: it must not be too intrusive. If the coach interrupted the student too often (i.e., every time the student made a mistake), the developers feared that the student would never learn to examine his/her own behavior and identify the causes of his/her mistakes. In addition, it was felt that an intrusive coach could spoil the fun of the game and destroy the learning opportunity. Thus, a computer coach must take careful account of two variables: when to interrupt the student's problem solving activity and what to say when it does. For the purpose of evaluating the overall theoretical orientation, student transcripts were reviewed for evidence of the coach's effectiveness in these two aspects. Several cases indicated shortcomings in the coach's ability. For example, one student was tutored twice on using different patterns and once on using parentheses. However, the student tried a new pattern only once (with poor results) and never used parentheses.

Figure V-2

Message Displayed on Screen	Annotations
You must have made a mistake. 2+2*1 is NOT 8 Would you like to change your expression? --> Yes	Student makes math error in number sentence.
You must have made a mistake. (2+2)*1 is NOT 7 Would you like to change your expression? --> Yes	Student makes second math error.
2+2-1=3	Subject tries a different number sentence which is correct but gives him a poor move. Student is then tutored on game strategy and given another chance to make a move.
You must have made a mistake. 2*(1-2) is NOT 0 Would you like to change your answer? --> yes	Student makes third math error.
You must have made a mistake. 2*(1-2)is NOT 2 Would you like to change your answer? --> yes	Student makes fourth math error.
You must have made a mistake. 2*(1-2) is NOT 1 It is -2 You don't get to move.	Fifth math error. Loses turn. Never found out what he misunderstood about his number sentences.

Another student, a good player in general, received tutoring and used hints on moving backwards and then tried for six minutes to make a number sentence to move backwards, as depicted in Figure 3. He finally lost his turn, never getting clarification on his error.

Student Modeling

Effective coaching requires (a) techniques for diagnosing what the student misunderstands (or does not know at all) and, (b) tutoring principles to guide WEST's advising and interrupting functions (Burton & Brown, 1982). WEST's authors felt it best not to disturb the flow of the game with testing to diagnose student weaknesses. Thus, the WEST coach must infer the student's misunderstandings from his/her game playing behavior. This inference can be a difficult problem. If a student does not encounter a chance to display a given skill, the computer cannot know whether s/he actually knows it. Hence, the computer relies upon a comparison of the student's skills used in a given situation to those used by an expert in the same situation. This technique, referred to as "differential modeling" by Burton and Brown, is fairly close to what is now commonly called an overlay approach. It is used to hypothesize what the student does not know or has not yet mastered (Burton & Brown, 1982).

Differential modeling in WEST requires that two tasks be performed by the coach with the computer expert:

1. Evaluate the student's current move in relation to the set of possible alternative moves an expert might have made in the same situation to determine the student's weaknesses.

Figure V-3

Lack of Effective Feedback

Message Displayed on Screen

Annotations

You might try moving backwards.
The numbers you should be able to make
are -25, -17, -16, -9, -8, -7, -6, -5, -1, 0, 1
I think moving -7 would be a good idea.
You can make -7 by the expression $7/(3-4)$
 $(4-3)/7$ All divisions must divide evenly.

Here are the 4 levels of hints he
requested.

Student tries but receives an
error message.

You must have made a mistake.
 $7/(3-4)$ is NOT 7
Would you like to change your expression?
--> Yes

Student tries another expression
but forgets negative sign. Gets
no instructive feedback.

You must have made a mistake.
 $7/(3-4)$ is NOT 7
Would you like to change your expression?
--> Yes

Tries again; same error.
No instructive feedback.

You must have made a mistake.
 $7+4*3$ is NOT 33
It is 19.
You don't get to move.

Gives up on suggested pattern.
Makes mistake due to order of
operations. No instructive feedback.
Loses turn.

2. Determine the underlying skills (e.g., using parentheses or choosing to bump) that went into the selection and composition of the student's move as well as each of the "better" moves of the expert, in order to construct a model of the student's knowledge.

Without preinstruction and pretesting, structure in the presentation is designed by student responses. While the authors did not establish a hierarchy among issues, a task analysis of the game suggests attention needs to be given to entry skill requirements and sequence of skill presentation. As can be noted in several of the examples above, many students had difficulty with the concept of negative numbers. In another situation, depicted in Figure 4, a student attempted to use division both before and after tutoring but lacked sufficient knowledge (possibly about basic facts and/or ordering numbers in a number sentence) and was unsuccessful. This student never tried using division again, despite the fact that patterns with division would have been optimal for many moves.

Burton and Brown (1982) acknowledge several sources of "noise" in their model: (1) The model cannot know which of several possible requisite skills for a given move were not understood by a student who makes an error; the model apportions blame more or less equally among all of the skills required for the missed better moves; hence, blame will almost certainly be placed on skills that are actually understood.

(2) Incompleteness in the set of skills represented results in the possibility that a student's reason for error will not be possible to detect and tutor. (3) Since students are not consistent, the differential

Figure V-4

Lack of Entry Skills

Message Displayed on Screen

Annotations

You must have made a mistake.
6+1/1 is NOT 0
Would you like to change your expression?
--> Yes

Student makes math error.

1+1/6 All divisions must divide evenly.

Student makes error and receives error message.

In addition, the student is tutored on division and given another another chance.

You must have made a mistake.
6-(1/1) is NOT 1
Would you like to change your expression?
--> Yes

After tutoring, student tries 3 times to use division, yet still doesn't understand.

You must have made a mistake.
6-(1/1) is NOT 0
Would you like to change your expression?
--> Yes

You must have made a mistake.
6-(1/1) is NOT 2
It is 5.
You don't get to move.

Loses turn and never tries division again in subsequent moves.

model may imply that they do not understand skills that they actually do but are too tired, distracted or bored to use them. (4) Finally, the differential model is cumulative and does not drop out earlier behavior; hence, even after a student learns a skill, his earlier weakness in that skill is still a part of his model. To try to compensate for this, Burton and Brown made the WEST coach "conservative" in that it does not tutor a skill more than once within a few moves. One consequence of this compensation is that corrective feedback is not always provided when it could be immediately useful. Again, the distinction between an informational and an instructional presentation is imperative. Feedback should adhere to specified instructional considerations -- appropriateness, immediacy, variety, and relevance. In addition, design of future WEST-type programs might consider allowing user control of tutoring frequency and content.

A serious educational limitation of the WEST student model is its inability to store information about student errors during attempts to formulate a move. This issue will be discussed later as it relates to knowledge representation and instructional strategy.

Knowledge Representation

A computer-based expert is required in WEST for the two modeling tasks of evaluating the student's move and determining the skills that went into it and all better moves. WEST's knowledge representation uses two different forms for these tasks: an opaque or black box model for the evaluation task and a transparent or articulate model for the determination of skills. The skill determination task requires the coach to consider the

individual skills involved in making moves to account for the student's failure to make a better move. In this case the developers judged that a transparent model was necessary.

The advantage of a transparent model is that it articulates skills in terms that match those of a human problem solver, making it easier for researchers to examine and manipulate. A major disadvantage is inefficiency. This model requires more than simple pattern-matching. For example, to ascertain whether a student understands parentheses the computer must determine not only whether or not they are present in the student's number sentence, but also whether or not they were necessary for the move or for an optimal move.

On the other hand, the evaluation task requires the expert to use only the result of its knowledge and reasoning strategies -- its better moves. Hence, the WEST creators consider the opaque model to be appropriate for this purpose. The use of the black box model for evaluating skills presents a serious limitation, however, because WEST cannot determine the student's reasoning behind his/her errors. There is no model for the algorithm the student uses to make an individual move. This is particularly troublesome in the case of math errors, some of which are not evident in the student model.

As stated previously, students in the current study frequently made math errors in constructing their number sentences to make their moves. At each error the computer told them they were wrong and asked them if they wanted to try again. Since the coach could not analyze their errors to provide detailed feedback, students made the same errors over and over.

Many made different errors trying to get the answer in different ways, and others made different errors while trying to make several completely different moves. An instructional expert would consider these various cases, analyze the patterns of errors, and then interact with the students accordingly. In addition, an intelligent tutor should be aware of the slowed pace caused by frustration or lack of help. Regarding interest and motivational factors, recognition of improved strategy and math skill would encourage students to continue trying.

One reason for non-adaptive tutoring is that WEST does not store behavior made during a single move and therefore does not include such information in its model of the student. Not only does the coach fail to alter its tutoring messages accordingly, the program does not record the number and type of errors made by the student. In several cases students made up to three mistakes during the course of a single turn but received no explanation of their errors and lost their turns. While they were given the answer to the equation, they were never shown or told why that answer was correct. Thus they endured several minutes of "failure" and were still left with unconstructive bugs. If they finally stumbled onto the correct answer, WEST's model of the student would reflect only the final success, not the trials and errors that preceded it. So the program fails in two critical areas: responding to errors for immediate learning and recording of errors for subsequent instruction.

Instructional Approach

To facilitate the student's learning from his/her mistakes, WEST's creators felt the coach's comments must be both relevant and memorable. To

accomplish this goal, Burton and Brown adopted an Issues and Example tutoring strategy. "Issues" somewhat similar to "rules" (Markle, 1983; Merrill, 1984), are abstract concepts used in the diagnostic process to identify what is relevant at a given moment of the game, i.e., what relevant skills the student fails to use. "Examples" provide concrete instances of these abstract concepts. Using both Issues and Examples in the information given to students increases the chance that the student will comprehend and integrate this tutoring into his/her knowledge base. See Figure 5 for an example of the WEST coach.

WEST provides tutoring on two of three possible levels of Issues: math skills (e.g., math facts, use of parentheses, correct order of operations), and WEST game playing skills (e.g., use of special moves, and trying to maximize the distance between the student and opponent). At present WEST does not tutor on the level of general game playing skills not specific to WEST (e.g., learning from one's opponent's moves) (Barr & Feigenbaum, 1982). Expanding the coach to deal with metacognition would be a particularly valuable future development in ICAI.

When the student makes a move, it is compared to that of an expert. The coach uses the evaluation component of each issue to create a list of Issues (skills) in which the student is weak. Using the expert's list of better moves, the coach employs the skill determiners (also referred to as Issue Recognizers) to determine a list of Issues that are illustrated by these better moves. The coach compares these two lists of Issues. If there are no Issues common to both lists, the coach infers that the reason for the student's error lies outside the set of Issues, and the coach says

Figure V-5

Example of Coaching from WEST

Perhaps you have forgotten that it's OK to rearrange the numbers.
You could get to be really good if you tried using other orderings.
One good example would be the expression:
 $(2*4)+1$,
which would have resulted in a TOWN!
YOU would have ended up at 70 with
the COMPUTER finishing up the turn at 54


Would you like to take your turn over?
=> ☐ YES ☐ NO

ecoach's turn

1 0 1
4 1 2
2 3 2 7 3
6 5 4

numbers are: 2 1 4

COACH



nothing to the student. If, on the other hand, the two lists have one or more Issues in common, the coach infers that at least one of these caused the student's error, so it selects an Issue along with an Example (a move that illustrates it). However, several students appeared to have "Issues" that went unidentified. For example, one student had three exchanges involving incorrect use of parentheses, lost a turn each time, and never had parentheses cited as an Issue. Other transcripts contained similar Issue identification concerns such as Figure 6.

When the coach decides to interrupt the student to provide feedback on an error, the coach activates a "speaker" attached to the appropriate Issue, which in turn presents a few lines of explanation. These text displays refer to the game move or number sentence the student used. While a rule-example format is used, other design features need to be improved. The displays are too simple in format and presentation. Lack of variety in style and print as well as long, closely spaced text make reading and retention difficult. Vocabulary and graphics could be better used to enhance attention, and interest and learning could be maximized by highlighting different kinds of information (key words and ideas) being presented. The long, run-on text displays of rather small print generally failed to hold student interest in the current study. It might be expected that students with low reading ability would encounter added difficulty. Thought units are not separated from each other and the display design lacks any consideration for individual learning styles. An instructional display needs to accommodate a range of individual differences and allow for learner control. Finally, the window of tutoring scrolls upward and does not allow the student to review the information presented.

Figure V-6

Unidentified Issue

Message Displayed on Screen	Annotation
<p>You must have made a mistake. $3*(2-1)$ is NOT 5 Would you like to change your expression? --> NO</p>	<p>Student makes error in use of parentheses.</p>
<p>You must have made a mistake. $3*(2-1)$ is NOT 5 It is 3. You don't get to move.</p>	<p>Loses turn.</p>
[Computer's move: $(3+2)*5=25$]	
<p>You must have made a mistake. $2+3*7$ is NOT 35 It is 23 You don't get to move.</p>	<p>Makes second error in using parentheses. Loses turn.</p>
[Computer's move: $1+(3/3)=2$]	
<p>$4*2+1=9$</p>	<p>WEST tutors on strategy of using shortcut despite fact student clearly need tutoring on parentheses. Offers student chance to re-take move.</p>
<p>A shortcut will advance you to its other end.</p>	<p>Student requests hint and is given one on shortcuts.</p>
<p>You must have made a mistake. $4+1*2$ is NOT 10 It is 6. You don't get to move.</p>	<p>Another math error on order of operations and use parentheses. Loses turn.</p>

A number of tutoring principles are built into WEST for deciding when it is appropriate to interrupt the game and which Issue should be selected. (See the Appendix for a complete list of Burton and Brown's tutoring principles). In the case where WEST must decide which of two or more competing issues to tutor, Burton and Brown have provided the user with a choice of how the coach will decide. One choice is the Focus strategy, in which the coach will tutor on a particular issue until it is mastered. The disadvantage of this approach is that the student may get quite bored with the coach's harping on this one weakness, particularly when all text for a given Issue is the same. The other choice the user has, which is the default system, is the Breadth strategy for selecting the issue on which to tutor. In this mode, the coach always selects an Issue that has not been recently discussed. The disadvantage of this mode is that an opportune moment for tutoring is often ignored.

Several additional tutoring principles follow the underlying philosophy that tutoring should be both relevant and memorable. Relevant comments are those pertaining to the Issue deemed the cause of the student's current error. Comments by the WEST coach are made memorable by providing an example and allowing the student to immediately practice the skill by repeating his/her turn. The value of both immediate feedback and appropriate practice are well documented in educational psychology. In the current research, however, it was observed that students often chose not to take their turn over again, i.e. to practice, after they had been tutored for an error. If a student did take a turn over, one practice problem seemed insufficient to learn a concept or develop an automatic response, evidenced by repeated errors and less than optimal moves.

For the many students who had problems with math skills involved in WEST (especially negative numbers and use of parentheses), it appeared that the tutoring information was insufficient to clear up their misunderstanding. In particular, there was little instructional content or examples and non-examples to clarify the concepts being tutored. Hence, students were not ready for the opportunity to practice. Of concern, too, is the fact that students did not receive tutoring for math errors made during attempts to formulate a move. They were only told they had made a mistake and given the opportunity to change their expression or answer. This was an ideal moment for relevant, memorable coaching. In fact, its absence seemed to leave many students unable to get past their math mistakes to truly engage the game. Clear specification of entry skills and effective skill presentation could help students avoid frustration.

A second major category of tutoring principles employed by WEST is concerned with maintaining the student's interest in the game. These include: not interrupting before the student has had a chance to discover the game for him/herself, not interrupting too often thereafter, congratulating the student on exceptional moves and identifying why they are good, and not forcing the student to retake his/her move after giving advice for an error. Giving congratulations for excellent moves seems a good use of positive reinforcement. Developers of WEST-like programs might consider that many students might profit from even more reinforcement, perhaps given for somewhat lower levels of achievement during the early stages of learning. More attention needs to be given to learner expectations and satisfaction. While WEST emphasized when not to

intervene, it does not address situations when it would be beneficial for the coach to interrupt -- for example, reiterating goals and objectives, encouraging intrinsic motivations, and suggesting attention to general skills and strategies of the game.

The program structure seems to proceed from two basic but questionable assumptions: (1) that the student does not want to be or should not be interrupted with game-pertinent information very often and, (2) that the computer should decide when and what to tutor the student. The present studies suggest that students without well-developed math skills might find additional tutoring less frustrating rather than more. In addition, it seems quite possible that students are sometimes aware of their need for tutoring and might profit from the opportunity to control the content and timing of tutoring as they can to some degree with the hinges.

Other tutoring principles in WEST involve increasing the student's chances of learning through the use of two techniques to guide discovery. The first is to provide several progressively helpful hints when the student asks for them. Note that hints are distinct from computer-initiated coaching. Four levels of hints are:

1. Tell student to consider a relevant weakness (e.g., "try using parentheses" or "try bumping")
2. Delineate the possible moves (e.g., "The numbers you should be able to make are ...")
3. Tell which move is optimal (e.g., "I think moving 0 would be a good idea.")
4. Give number sentence for making that move. ("You can make 0 by the expression $3-(2+1)$."

The concept of offering graduated hints is educationally sound. However, lack of sufficient embedded direction and guidance for using hints caused unnecessary frustration for several subjects in the present study. WEST does not explain the hint "system" to students; they must discover it. In fact, in this study many students did not ask for additional hints after the first one and simply persisted in their errors without making progress. During the final interview, one student's responses suggested embarrassment as a cause for failing to ask for hints.

Regarding content, the hints seemed more often related to game moves than to math skills. When a student constructed an incorrect number sentence, neither the coach nor the hints addressed the math error.

Unfortunately there does not appear to be a mechanism in the program to respond to a student's pattern of asking for hints. For example, several students used hints on almost every turn while some students never used them. Both conditions may warrant comments from the coach. Future development of WEST-type programs might include hints that are both informative and instructional, sufficient guidelines for their effective use, and a record in the student model of the pattern of requests for help.

Another technique WEST uses for increasing students' learning is that the computer always plays an optimal game. Brown and Burton point out that one of the best metaskills a student can learn from WEST is to watch what your opponent is doing, especially if you are losing. They feel that watching an expert play is better than watching someone who makes mistakes. In the current research study, the children tended not to be able to infer the best game strategy from watching the computer for

three-quarters of an hour. Many students were simply not facile enough with number sentences to be able to turn their attention to the opponent's strategy. Transcripts suggested that students' number sentences were generally unaffected by the computer's modeling of number sentences during its move. It may be hypothesized that the ability to learn from observation is a function of the difficulty level of the skills required to play the game. In WEST's case, the computer's complete game strategy is not readily apparent from an hour's observation.

A particularly fertile area for future research and development is how to enhance subjects' learning from observing the opponent's play (and how to make this process most efficient.) When playing WEST the student does not see how the computer decided what to do, only the resulting number sentence and move. Unfortunately, due to the use of the black box expert, it is not currently possible to display the logic or problem solving approach of the computer's move. Designers of future WEST-like programs might consider the value of using articulate knowledge representations for the expert part of the program in order to have the option of providing this information for the learners.

It may also be important to make the distinction between the effects of watching someone play and actually playing against that person (or machine). Constantly losing to an expert opponent can be boring and disheartening, a phenomenon observed in the current research. This problem was not overlooked by WEST's authors. One of the principles they used in WEST was adjusting the level of play when the student was losing consistently. However, this principle conflicts with always having the

computer play the optimal game. Their solution was to adjust the level of play after a certain number of consistent errors by varying the computer's spinner numbers in relation to the quality of the player. During the current research study many students did get discouraged by losing frequently during the 45 minutes they interacted with WEST. It was not evident whether the computer's spinners were ever adjusted as there were very few student wins.

Furthermore, it was noted that some of the frustration at failure was not successfully addressed by the possibility of the computer having "bad" spinner numbers. In one example, shown in Figure 7, a student encountered great frustration during the first five moves of her first game. On every move the computer consistently bumped her back to the beginning. In one move the computer's bumping her actually resulted in moving it less optimally than another possible move could have done. In interviews after the game, several students noted their frustration in trying to avoid being bumped by the computer and expressed some hostility when they said they would tell a friend to "try to bump that guy!". The computer certainly succeeded in getting some of them to focus on this issue; however, they did not necessarily learn much about how to bump and to avoid being bumped.

Future research might investigate the effects of lowering the ability of the expert at first rather than after frequent losses. Once the game engaged the student's interest and provided some self-confidence, it would be reasonable to gradually raise the level of the expert's play.

Another WEST principle was employed to make the gaming environment more interesting: if the student makes a potentially careless error, be

Figure V-7

Message Displayed on Screen	Excess Bumping Causes Frustration Annotations
$(3*1)+1=4$	Student moves to "4".
[Computer's move: $(1*2)+2=4$]	Computer moves to "4"; bumps subjects back to "0" leaving computer ahead by 4 spaces. Computer <u>could</u> have moved to "5" & taken shortcut to "13".
$(6*1)+1=7$	Student moves to "7".
[Computer's move: $6/(1+1)=3$]	Computer moves to "7", bumps student back to "0".
"Maybe you should look for towns." $(3*3)-0=9$	Student requests and receives hint. Student moves to "9"
[Computer's move: $3-(2/2)=2$]	Computer moves to "9", bumps student back to "0".
You must have made a mistake. $(4+1)*3$ is NOT 2 Would you like to change your expression? => Yes	Student makes math error.
$(4+1)*3=15$	Student moves to "15" and is then tutored on "bumping".
[Computer's move: $(3*2)/1=6$]	Computer moves to "15", bumps subject back to "0".
$(6+1)*3=21$	Student moves to "21". Did not learn to bump from being tutored and bumped several times.
[Computer's move: $(2*4)-2=6$]	Computer moves to "21", bumps student back to "0" again, 5th time!

forgiving, but provide explicit commentary in case it was not just careless. The WEST system contains diagnostic routines for many typical errors a student might make, such as giving the end position of a move as the value of his/her number sentence. However, very few such comments appeared to be activated during the games observed in this study. Lack of such information added to student frustration in many instances.

Burton and Brown mention that while these tutoring principles are currently compiled into the system, they need to be represented articulately so they can be modified and their effects observed. An additional improvement for future consideration is the incorporation of fundamental instructional design factors with the tutoring principles. These should include a more complete knowledge base and the ability to alter teaching strategies.

The developers' treatment of when and what to tutor lacked certain instructional components, but their considerations for how to tutor are more severely limited. As mentioned elsewhere in the report the action and graphics on the screen are quite simplistic and text displays are inadequate. The conversational nature of the text displays is important for engaging student thought, but students are never involved in constructing conceptual responses or articulating concepts and strategies. No explicit guidance in learning strategy is provided. In addition, complete reliance on written text on screen makes success very dependent on students' reading abilities. Finally, the math skills/game format did not seem relevant to some students. For example, one student commented that some of the WEST math skills would not be used in "real math"!

Model of ICAI Development

To what extent does WEST serve as a model of ICAI development? A model ICAI system should exhibit expertise in three components: knowledge representation, student model, and instructional strategies, and WEST clearly has each of these. However, each of these components in WEST is limited in some way, and future WEST-like programs could benefit from avoiding these limitations.

Regarding WEST's knowledge representation, the subject matter expertise is incomplete. It is not apparent that a task analysis had been done to determine the prerequisite, developmental, and maintenance level skills required to play the game.

The WEST student model includes lots of information, but it is not particularly helpful in planning subsequent instruction for the student. It is more a report of behavior as opposed to a diagnostic and prescriptive document that could be educationally useful.

A sophisticated instructional system (CAI, ICAI, or human) selects, sequences and presents information in very adaptive ways. An ICAI system should be able to make decisions about these three processes based on the student model it builds and on its level of instructional expertise. That instructional expertise is lacking several critical features in WEST, such as explicitly stated objectives and criteria, and appropriate and instructive feedback to the student.

Army Needs

To what extent does WEST exhibit instructional content and features potentially useful to future Army applications?

There are several aspects of Army applications that are relevant to considerations of ICAI program usefulness. The Army is faced with the need to continually instruct a large number of people, who have a wide range of abilities and needs, in a broad spectrum of skills from the very basic to highly complex skills necessary in rapidly changing technical systems. Furthermore, this instruction needs to be replicable across many, varied and scattered locations. And finally, high quality training with a high success rate is often imperative despite limited resources of time, money, and human experts.

Perhaps with instructional enhancement WEST could be useful in certain Army applications involving basic skills, strategy and problem solving. But to more broadly address Army needs, it is more appropriate to consider WEST-like ICAI programs. The features in this type of program that may be useful to the Army include:

- a. providing interest and motivation to a learning situation through a game environment,
- b. addressing individual needs and abilities through a self-paced, student-controlled learning situation,
- c. addressing the multiple content considerations through a well-developed issues and examples paradigm, and
- d. providing effective means for teaching problem solving processes as well as factual and technical information through a well-designed coaching system.

Summary

In summary, WEST demonstrates both the complexity and potential of intelligent computer assisted instruction. However, several serious concerns have been articulated regarding the need for instructional enhancement of WEST and WEST-like programs. These include: (1) complete documentation of the program with user manual and clear exposition of learning objectives; (2) clear specification of entry skills; (3) provision of directions and guidance for using program features such as hints; (4) extension of the knowledge representation to include additional requisite skills; (5) articulate representation of the computer's moves along with the ability to manipulate the display of this logic to the learner to enhance observational learning; (6) inclusion of a model for student behavior to diagnose math bugs during attempts to formulate a single move; (7) use of some diagnostic questioning of the learner to supplement the inference process; (8) feedback that is more immediate, detailed and instructive with some learner control of content and timing; (9) graphics and instructional content with greater learner appeal and the capability of enhancing attention and retention; and (10) user ability to regulate the difficulty level of the game.

The theoretical strength of WEST is the attempt to help students benefit from their errors. The practical limitations of this program are the inaccurate analysis of "bugs" and the inadequate instructional treatment.

EFFECTIVENESS STUDIES

Evaluation Questions:

3. What are the learning outcomes for students? To what extent do learners achieve project goals? Do students with different background characteristics profit differentially from exposure to WEST? To what extent does it create unanticipated outcomes, positive or negative?

Two separate effectiveness studies were conducted using WEST. The first was conducted off-line, prior to the delayed equipment arrival. Its purpose was to explore preliminary criterion measure development and to assess the effects of stronger instructional interventions as a precursor to the experimental contrasts involving the playing of WEST. Following descriptions of design, methods, and analyses of these studies, general conclusions and recommendations will be presented.

WEST Study 1

Method

Design and Subjects

The experimental design utilized two treatment groups and one control group. Twenty-one students from a fourth-to-sixth grade class at the Corinne A. Seeds University Elementary School* at UCLA were selected to

* The UES student population is planned to be representative of the student population of the U.S. in ability level, ethnic derivations and socioeconomic levels.

participate in the study on the basis of good math ability as identified by their teachers. Approximately half were boys, half were girls. Within each sex, students were randomly assigned to the three groups. Six students were in the control group, seven were in the "math key" treatment, and eight were in the "strategy key" treatment group.

Materials

Due to the failure of the AI machine to arrive until the last few months of the project (as already noted), an off-line board game version of WEST was used in a pilot study to investigate the effects of different specific cues on learners' game playing ability and subsequent math and strategy measures. The WEST board game was an exact copy of the WEST computer display, including spinners and the board. See Figure 8.

Two "keys" were developed for the two experimental conditions in this study. The "Math Key" consisted of a cardboard cue card about 8 by 10 inches with three brief statement of math concepts that would be helpful in playing WEST: variety of number sentences, order of operations, and grouping with parentheses. The "Strategy Key" consisted of a similar cue card with three brief statements of game strategy concepts that would be helpful in playing WEST: strategies to choose, decisions to make, and the move to pick. See Figures 9 and 10 for details of both key cards.

The keys were developed from a task analysis of playing WEST, and they were intended to serve as brief advance organizers for directing subjects' attention to important elements in playing the game successfully and as prompts available during the actual play of the game. The game materials and keys were piloted with a few students ahead of the time and a few minor

Figure V-8

WEST Board Game

Stagecoach's turn		
The numbers are: 1 4 4		
1	4	4
+	-	*
/	()
bs	ok	Hint
YOUR MOVE:		

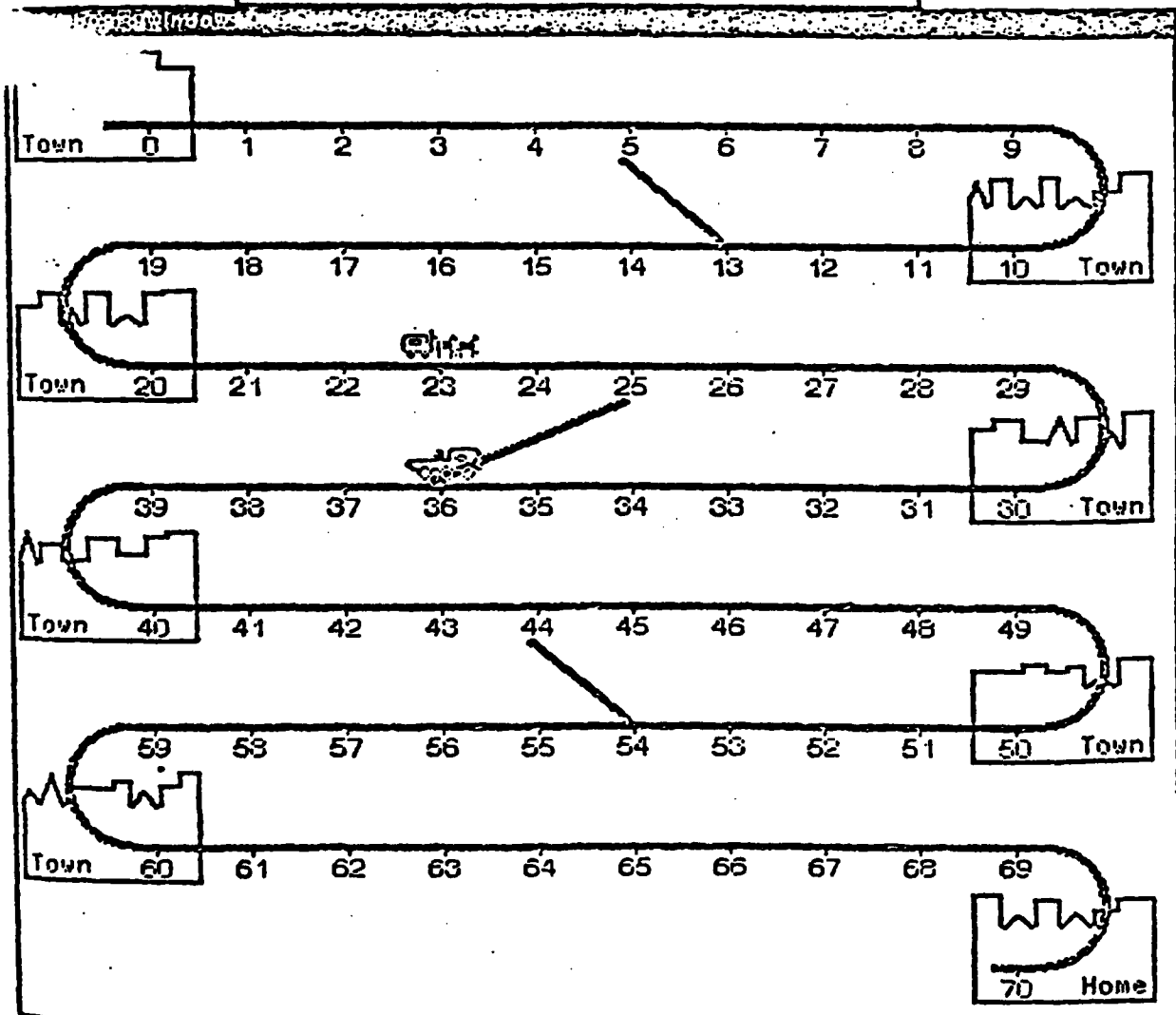


Figure V-9

WEST 1 Math Key Card

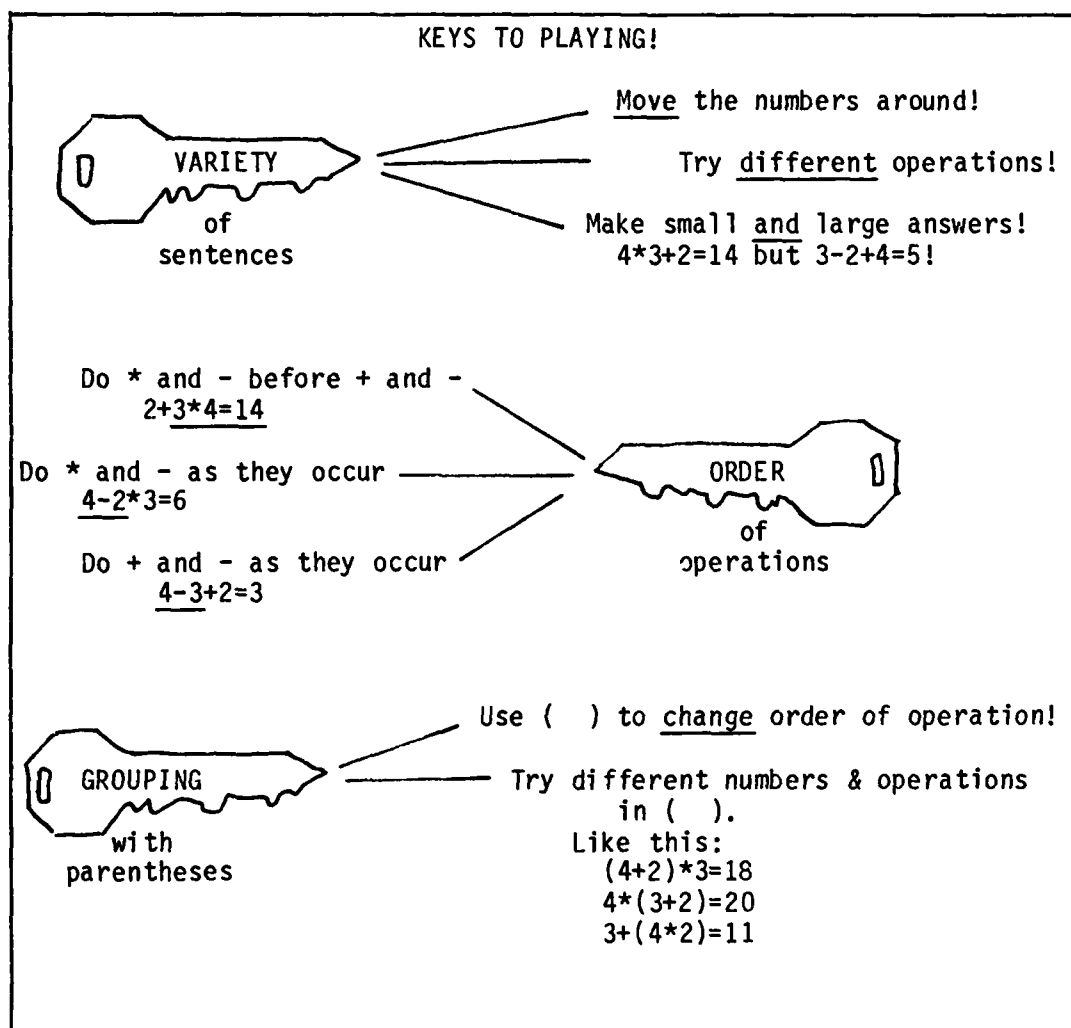
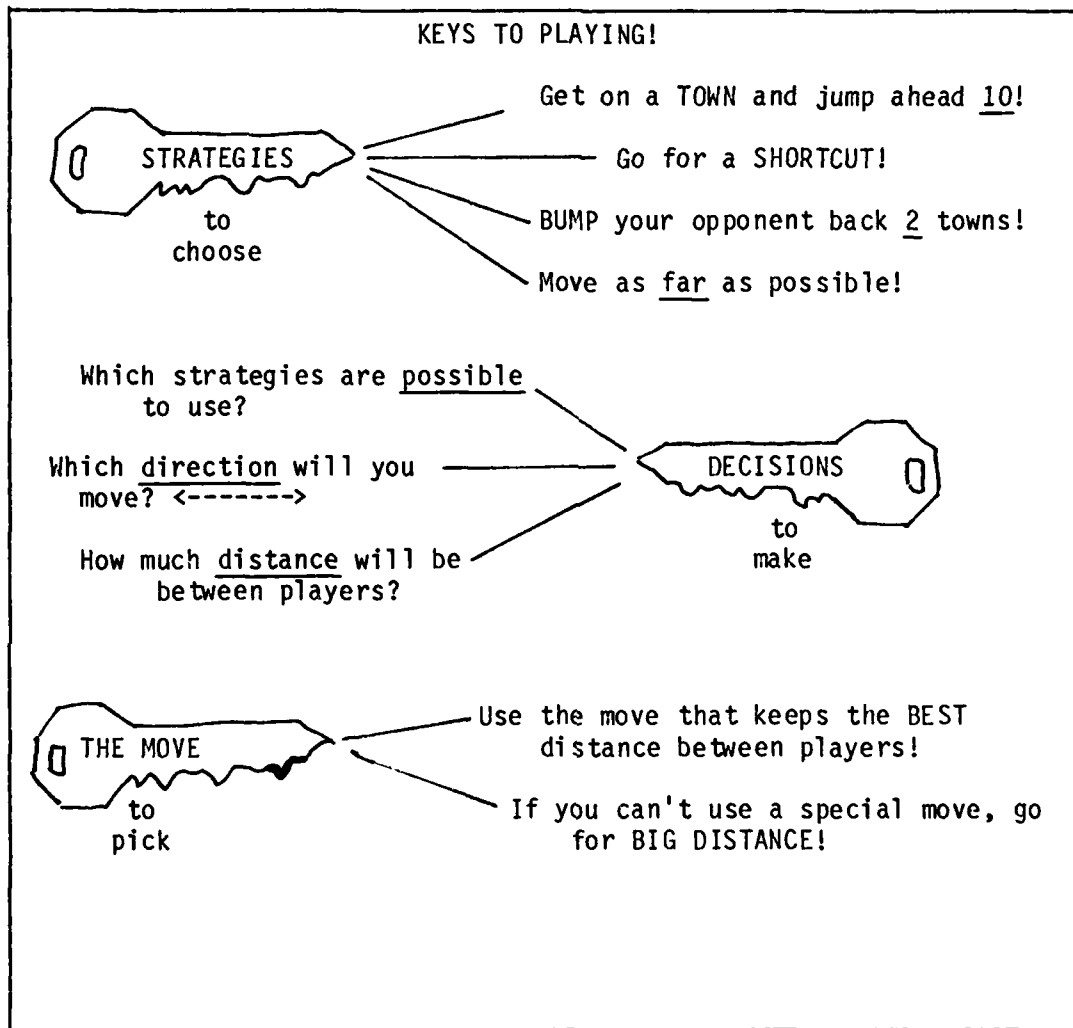


Figure V-10

WEST 1 Strategy Key Card



revisions were made.

Measures

Questionnaire. Subjects were given a 2-page questionnaire consisting of 10 questions about playing games with answers based on a 5-point scale.

For example,

1. How much do you like to receive help during a game?

not much		some		a lot
1	2	3	4	5

Math Pretest. The 6-page Math Pretest consisted of 139 items covering:

- o addition and subtraction facts
- o multiplication and division facts
- o families of facts
- o factors of numbers
- o number sentences to solve in the WEST format -- open-ended and multiple choice

Dependent Measures. Four dependent measures were used: Observational Ratings, the Math Posttest, the Strategy Posttest, and an Interview.

A trained observer rated 10 aspects of subjects' behavior, including effort, enthusiasm, and ability, during the game on a 4-point scale, the

Observational Ratings. A sample question follows.

1. The student was enthusiastic:

rarely or not at all			most of the time
1	2	3	4

The Math Posttest consisted of 16 number sentences to solve (some with parentheses), four items in which the subject was to construct a number sentence with given numbers to equal the largest possible answer, seven number sentences to write to obtain given results, three different number sentences to construct to obtain a single result, and an item in which to generate number sentences to obtain as many different results as possible. Sample items follow.

1. $(1 - 1) * 5 = \underline{\hspace{2cm}}$
2. Write a number sentence with the numbers 1, 2, 5 to equal the largest possible answer
3. Use 2, 1, and 6 in number sentences to get as many different answers as you can. Use the WEST rules.

The Strategy Posttest consisted of nine items, each showing a different game board position and spinners for which the subject was to write his/her desired move, write a number sentence to obtain the result necessary to make that move, and circle the place where the stagecoach would land.

The Interview consisted of thirteen open-ended questions to elicit attitudes toward the game and the keys used. For example,

1. Did you learn anything from playing the game? What?
2. What was most helpful about the "Keys"?

Procedure

Each subject was escorted from the classroom to a nearby small room where s/he was interviewed briefly about attitudes towards games (the Questionnaire) and then given the Math Pretest, which took about 15 minutes.

The student was given a copy of the game directions to read aloud. After each numbered item the experimenter asked if s/he understood and gave reading assistance as necessary. The experimenter answered all questions that pertained to the rules. "What if ..." questions were answered by referring to the appropriate rule or by saying that it would become clear during the practice game. For experimental subjects, the experimenter presented the appropriate key card and said, "These are some helpful hints for playing the game. Read each out loud." After reading each one, the student was asked, "Do you have any questions?" The experimenter emphasized that the Keys could be used at any time during the game (they were displayed in convenient view throughout the games), but that no explanations could be given.

The model for play was:

1. spin each spinner
2. record the numbers on the worksheet
3. work out number sentence
4. show number sentence to opponent for agreement on result.

Students were allowed to play WEST for 45 minutes. While more interaction time with WEST would have been desirable for optimal learning, students could only miss about two hours of class time, including directions, play, testing and interview.

Results: WEST 1

One-way analyses of variance revealed no significant differences among the three treatment groups prior to playing WEST in either their questionnaire responses about attitudes towards games, math and computers, or in their math skills as measured by the Pretest. After receiving instructional treatment playing WEST, the three experimental groups were still not significantly different as evidenced by one-way analyses of variance of math and strategy posttests.

The Observational Ratings piloted during this study included ratings on ten aspects of behavior on a 4-point scale, and included enthusiasm, effort, general behavior, ability at the game and math skills. The scale seemed to provide some useful information; however, a few changes were made for WEST 2. Items were put on a 5-point scale to spread out the subjects' behavior. A few items were reworded to simplify the form. A tally was added: the number of times the student commented and asked questions.

An analysis of the Strategy Posttest by Math Pretest errors revealed that two strategy items did not function very well, i.e., students could select the optimum strategy even though they made many math errors in math subscales that appeared relevant to the strategy problem. A third item was dropped from the test for the second study to conserve time and because it did not add substantial information about the students' game playing ability.

There were no apparent treatment group differences in the final interview results. For example, students in each group noted that they had learned a variety of ways to combine three given numbers in number

sentences to produce different answers. Likewise, at least one student in each group mentioned that the hardest part of the game was to avoid being bumped by the computer. Since students seemed able to respond to all of the questions, all items pertaining to the computer version of WEST were retained for the second study.

WEST Study 2

Method

The major puprose of this online study was to investigate the relative effects of playing WEST alone or with instructional treatments to enhance math and game playing skills addressed by WEST. A second purpose was to assess student reactions to the WEST program.

Design and Subjects

The experimental design utilized two treatment groups and one control group. Thirty potential subjects from a fourth-to-sixth grade class at the Corinne A. Seeds University Elementary School at UCLA were matched in groups of three on the basis of age and sex. Due to time limitations resulting from delayed arrival of the computer for WEST only seven of the ten groups were randomly selected to participate, and the subjects in each group were then randomly assigned to the three treatment groups. Attrition reduced the number of subjects who participated fully to 18, nine boys and nine girls. Five subjects remained in the control group, six in the math training group, and seven in the strategy and math training group. The poorest math students (those who lacked proficiency with basic number facts) had been screened from the study ahead of time by their teachers to avoid their being too frustrated with the task.

Materials

Two self-instructional training booklets were developed for the experimental conditions in this study. One focused on math skills useful in playing WEST. The other focused primarily on relevant game strategy

with somewhat less emphasis on math skills. Both booklets took about 20 minutes to finish and were approximately 22 pages long. The first two pages of both booklets provided a general introduction to WEST and its special symbols. (E.g., in WEST, / means division and * means multiplication.) Both booklets are described below and also appear in the Appendix.

Math Training Booklet. This booklet presented "four important math ideas" that were similar to the math skills addressed in the first study. These "ideas" were:

1. You must use correct order of operations.
2. You can use parentheses in number sentences.
3. Some number sentences have negative answers.
4. Some number sentences follow patterns (demonstrates the pattern for obtaining the largest possible result using WEST rules.)

The first two "ideas" were drawn directly from the "keys" used in WEST 1. The third "idea" was developed after observing in the WEST 1 study that students were largely unaware of the mathematical procedure for moving backwards (negative numbers). The math key "variety of sentences" used in the first study was judged to contain elements of game strategy as well as math, so this key was revised to create the fourth math "idea" for the WEST 2 study.

In this study the "ideas" were presented in an instructional format involving presentation of the "ideas", example, problem, feedback, more problems, feedback, clarification of details, and summary. Each "idea" was

presented on different colored pages to enhance interest and retention. The math involved in each idea was always presented within the context of WEST rules (i.e., given three spinner numbers, they must be combined in a number sentence using each number exactly once and using 2 different operations.) So as to isolate this treatment from the Strategy-and-Math Treatment, no mention was made of why certain results would be strategically valuable in the game. At the end of the booklet a chart summarizing the 4 ideas was presented and seven practice problems were given.

Strategy and Math Training Booklet. This booklet presented four "game ideas" designed to help the student to play WEST well that were revised versions of the strategy keys used in WEST 1:

1. Look for your options (describes 4 special moves in WEST)
2. Figure out how many spaces you need for each option
3. Use math rules to find which numbers are possible
4. If you can't win, get the best distance possible between you and your opponent.

These ideas present a general problem solving approach to the game, which seemed to be lacking in most students' approach to the WEST board game in the first study despite the availability of the "keys". This training booklet was designed to be a brief yet more comprehensive approach to game strategy than had been attempted in the WEST 1 study.

Three of the four "math ideas" were presented in the Strategy and Math Booklet as part of the section on using math rules to find which numbers are possible. Negative numbers were omitted in order to keep the

treatments about the same length. (Note that all subjects were told in the game directions that negative numbers were acceptable solutions and would result in moving backwards.)

The Strategy and Math Booklet also presented its main ideas on color coded pages in the same instructional order used in the math booklet. The booklet ended with a summary of the ideas covered and several final practice questions.

Measures

WEST 2 was concerned with the same general variables as WEST 1: attitudes, math skills and game strategies. The availability of the computer made it possible to investigate response time, "quality" of moves, coaching incidents, requests for hints, and use of parentheses.

Background measures. March, 1985 CTBS/s level 2 math subscale scores in computation and concepts were obtained for all subjects. In addition, subjects were given a Pre-Game Interview, derived from one used in the first study, which consisted of nine questions on a 5-point scale to elicit their attitudes toward games, math, and computers.

Math pretest. A 2-page, 5 minute timed version of the WEST 1 Math Pretest was given prior to playing WEST. Revisions were made for WEST 2 on the basis of a content analysis of the WEST 1 version and the need to reduce student testing time in order to maximize time on the computer.

Individual Difference Measures. Several measures of students' behavior were taken during the game playing period. On the Observational Ratings Form a tally was kept by a trained observer of the number of questions and comments each subject made and ratings on a 5-point scale were made of students' enthusiasm, effort, ability and behavior.

The computer was programmed to record a number of other variables related to game playing: (a) the response time in seconds for each move; (b) the quality rating of each move as determined by the WEST program (best, good, fair, poor); (c) the number of coaching incidents each subject encountered; (d) the number of hints requested by the subject; (e) the length of the game in minutes and seconds; (f) the way in which the student used parentheses in number sentences (necessary and correct, unnecessary but correct, and incorrect or not used when necessary).

Dependent measures. Three dependent measures similar to those in WEST 1 were used: a Math Posttest, a Strategy Posttest, and Post-Game Interview.

The Math Posttest from WEST 1 was slightly revised. Four new items were added to assess students' ability to transfer their skills to work with larger numbers. Following are several sample items.

1. Solve this number sentence: $7 * 0 - 3 = \underline{\hspace{2cm}}$
2. Write a number sentence with these numbers that equals the LARGEST POSSIBLE ANSWER. Change the order of the numbers and use parentheses as needed. All WEST rules apply (do not repeat operations, use each number once, no fractional answers.)

1, 2, 5

3. Write a number sentence for ...

2, 2, 7 to equal 12

4. Use 2, 1, and 6 in number sentences to get as many different answers as you can. Use only the numbers 2, 1, and 6. Use WEST rules!

5. What is the largest possible number you can make using 12, 20, and 9 in a numer sentence?

_____ = _____

The Game Strategy Posttest used in WEST 2 contained 5 sample WEST moves with several open-ended and multiple-choice questions about each move to test each of the four problem solving steps presented in the Strategy and Math Training Booklet. Sample items from the WEST 2 Strategy Posttest follow.

1. Look at Game board #2. What does the stagecoach need in order to BUMP the train?

a) 4 b) 5

2. Look at the spinners in Game board #3 and decide if the stagecoach can make this move. Show a number sentence for each possible move.

3 is a) possible _____ = 3

b) impossible

3. The stagecoach in Game #4 can make these moves: 3 4 8 12

Which move is BEST? _____

Why? _____

What equation would you use for this move? _____ = _____

The Post-Game Interview was similar to that used in WEST 1 and included open-ended questions about the subjects' attitudes toward WEST, the coach, and the training booklets, for example:

1. How did you like the "coach"? Did it help you? How? Did it bother you or interfere?

2. What three things about the game would you tell a friend to help him/her to play the game?

Procedure

Each subject was escorted from the classroom to an adjacent quiet patio where s/he was interviewed briefly about attitudes towards games, math and computers, and then given the 5-minute Math Pretest. Experimental subjects were then given a self-instructional training booklet and told to work through it, which took about 20-25 minutes. Experimenters were trained to give standard, non-instructional replies to subjects' questions. Six of the subjects received training in math skills deemed important in playing WEST. Seven of the subjects received training in both math and game strategy skills (in a single booklet).

The following day experimental subjects were escorted from their class to a small room in a separate building on campus to play WEST on the computer. Control subjects were interviewed and pretested immediately prior to playing the game. Due to a problem with the computer at the last minute, subjects received game directions from an experimenter rather than from the computer monitor, as is normally done with WEST. These directions were exactly the same as those given by the computer.

Subjects were allowed to play WEST for approximately 45 minutes each, with only one subject in the computer room at a time. As in WEST 1, longer interaction with WEST would have been preferable but was not possible under the circumstances. During play the experimenter provided very little verbal input as the computer is programmed to interact with the subject via

the game board, and hints from the coach are displayed on the monitor at appropriate times. No "instructional" input was provided by the experimenter. A single experimenter ran all subjects. It was noted informally afterwards that Control subjects in particular seemed most frustrated by the game as might be expected since they had had no prior introduction to it.

The day after playing WEST each subject was given the math and strategy posttests in a small group of 2-4 subjects during a 45 minute period. After the posttests each subject was interviewed individually about the game and training experience before returning to class.

Results: WEST 2

The students participating in WEST 2 exhibited a broad range of math ability as measured by the CTBS math subscales in computation and concepts, from a low of the 26th percentile to a high of the 99th percentile. See Table 2 for subjects' mean scores on the CTBS tests and three other measures used in this study.

Prior to playing WEST, the three treatment groups showed no significant differences in their attitudes toward math, games and computers (as measured by the Pre-Game Interview items), nor in math computation and math concepts (measured by two CTBS subscales), or math skills required in playing WEST (measured by the Math Pretest). See Table 3 for ANOVA results for the math measures.

The Pre-Game Interview included nine questions about attitudes toward math, games and computers, all on a 5-point scale with 5 being most positive. Students' ratings on this instrument averaged approximately in the center of each scale for all but two of the questions. The majority felt they were quite good at playing games, and that it is important to read game directions. Correlations between the Pre-Game interview scales tended to be very small (with the notable exception of a high correlation between scales assessing the perceived value of knowing about the game beforehand and reading its directions, $r=.85$, $p<.001$).

The Math Pretest assessed basic skills, fact families, and number sentences. Correlations were generally positive between these subscales.

Of primary interest in this study was the possible effect of training in math or in math and strategy skills on performance during the game of

Table V-2

Means and (SD) for Math and Strategy Measures

Treatment Group	n	CTBS		Math		Strategy
		Math Computation Percentiles	Math Concepts Percentiles	Pre	Post	Post
Math training	6	71.83 (21.56)	69.00 (21.66)	35.67 (1.51)	13.50 (7.64)	11.17 (5.04)
Strategy training	7	70.71 (20.01)	65.86 (24.94)	34.71 (2.63)	14.86 (9.34)	12.14 (4.06)
Control	5	54.60 (27.82)	68.60 (12.18)	36.40 (0.55)	12.40 (3.51)	11.60 (2.88)
Overall	18	66.61 (22.78)	67.67 (19.87)	35.50 (1.91)	13.72 (7.20)	11.67 (3.93)

Table V-3

ANOVAs for Group Effect
Pre-Treatment Differences

	SS	DF	MS	F	P
CTBS math computation	1002.82	2	501.41	0.91	n.s.
CTBS math concepts	37.94	2	18.97	0.04	n.s.
Math Pretest	8.54	2	4.77	1.14	n.s.

WEST, on subsequent performance on math and strategy posttests, and in attitudes and opinions reported after the game. One-way analyses of variance revealed no significant differences among treatment groups in regard to number of questions asked during the game, the Math Posttest scores or the Strategy Posttest scores. See Table 4.

Sixteen of the 18 participants left an internal record of every move,* and the West software generated extensive summaries regarding the machine's responses, commentaries, and ratings of each subject's experience. Nine summary statistics are presented in Table 5: the average time required for each response, the minimum and maximum times in which a given subject completed a response, the length of a game, the number of hints the subject requested from the computer, the percent time of parentheses used in constructing equations were rated in each of three categories, and the number of coaching incidents which the computer gave to the subject during a game.

The shortest complete response per move made by any participant was 7 seconds long, while the longest time required was over 6 minutes. On average, each student spent a little over one minute in constructing a move, and completed a game every fifteen minutes. The use of parentheses evenly divided into correct, optional, and wrong, with at least one student never able to use parentheses correctly.

* Two subjects' records were not recoverable.

Table V-4

ANOVAs for Group Effect
Post-Treatment Differences

	SS	DF	MS	F	P
Tally of questions asked	8.32	2	4.16	3.49	n.s.
Math Posttest	18.05	2	9.03	0.51	n.s.
Strategy Posttest	3.11	2	1.56	0.10	n.s.

Table V-5

Performance Transcripts - Means

Variable	Mean	s.d.
Response time per move (seconds) averaged within subject**	71.73	57.57
Response time minimum (seconds)	19.94	10.31
Response time maximum (seconds)	164.44	87.00
Length of game (minutes) averaged within subject	14.91	11.91
Number of hints requested per game, averaged within subject	2.96	3.81
Use of parentheses: percent		
"correct and necessary"	30.37	26.16
"correct but optional"	31.00	26.51
"not correct"	38.62	30.62
Number of coaching incidents per game, averaged within subject	1.29	0.62

Notes: ** response time in seconds for individual to complete his/her average move.

Response time measures are well-correlated with one another (Table 6) and are also significantly related to the CTBS math computation subscale (Table 7). However, the relationship to the CTBS math concepts score is negligible. More able participants worked faster, and asked for more hints. The number of coaching incidents is related to the math pretest and posttest scores: more able participants experienced less coaching.

Every response made by a participant in this study was rated by the WEST program. Across all 229 separate moves made by the 16 subjects whose games were recorded, "best" and "good" ratings were given to almost two thirds of the moves made, as depicted in Table 8.

Observations made by an observer during the game included ratings on nine items (each on a 5-point scale with 5 being "a lot" or "excellent"). The items covered enthusiasm, effort, general behavior, ability at the game and math required by the game. Results from these observations were generally at midrange on average, except that amount of effort shown by the student was high in almost every instance. Some interview items were highly correlated with one another: enthusiasm and effort, number of comments and number of questions, improvement in ability during the course of the game with quality of math and game playing exhibited.

A series of analyses of variance were conducted to examine possible treatment group and sex differences in the various pre-test and post-test measures. Only Item 4b of the postgame interview proved to demonstrate a significant difference. The item is a tally of the number of questions asked by the student during the time she/he spent playing WEST on the Xerox 1108. Girls were significantly less likely to ask questions than boys, as

Table V-6

Response Time Per Move Intercorrelations

		average	min	max	length
Response time:	average	1.00	.80*	.84*	.91*
	min		1.00	.60*	.61*
	max			1.00	.76*
	length of game				1.00

Notes: * $p < .05$

Table V-7

Performance Transcript Correlations with Various Measures

	Math pretest	Math posttest	CTBS computation	CTBS concepts	Strategy posttest
Response time avg.	.12	.20	-.56*	-.01	.22
min.	-.05	.02	-.06*	-.20	.20
max.	.09	.26	-.57*	.00	.14
game length	.31	.20	-.25	-.07	.19
Number of hints	.18	.09	.54*	.16	.36
Parens. correct.	.28	.25	-.42	.25	.53
optional	-.40	-.13	-.16	-.37	-.27
not correct	.10	-.10	-.21	.12	-.22
Number of coaching incidents	-.57*	-.64*	-.44	-.45	-.68*

Notes: * $p < .05$

Table V-8

Response Time vs Rating of Move

229 moves made:	%	Mean (secs)	s.d.
"Best move"	38	79.23	69.30
"Good move"	24	63.91	103.89
"Fair move"	22	42.98	40.30
"Poor move"	14	54.15	34.68
"Other"	2	27.50	10.12

shown in Table 9. No other significant differences between treatment groups or between the sexes was found.

Items in the Math Prettest showed uniformly positive intercorrelations (average $r=.42$). Overall scores correlated positively ($r=.67$ and $.82$) to Math Computation and Math Concepts from the CTBS.

The posttest of game strategy skills consisted of five sample WEST moves with open-ended and multiple-choice questions about each move. All strategy simulations but the third showed substantial correlation to skills at solving or writing number sentences. The last simulation is well related to the Math Concepts subscale of the CTBS ($r=.64$).

Across the various measures these data show a modest positive relationship between Math Pretest and Posttest ($r=.42$), a modest positive relationship between Math Pretest and Strategy Posttest ($r=.55$), and a large positive relationship between Math Posttest total score with Strategy Posttest total score ($r=.77$). See Table 10.

The Post-Game Interview consisted of seven open-ended questions about the game experience. Nine out of 18 students (clearly the most frequent response) said that the hardest part of the game was making number sentences. The easiest part of the game for five students was math facts; for four others, using the mouse. Most students (11 out of 18) felt they learned something from playing the game; of these, 9 said it was something related to math (e.g. getting better at parentheses). Most (13 out of 15 who answered) also felt they liked the coach in WEST. Note that "coach" here refers to both hints and true coaching as students were unable to detect the difference. Ten felt it was helpful and did not bother them.

Table V-9
Means and ANOVA for Questions Asked

Means:

Treatment	Strategy/Math Training	Math Training	Control
Males	4.00	3.00	4.50
Females	2.75	1.67	4.00

Analysis of variance

	SS	df	MS	F	P
Treatment	8.32	2	4.16	3.49	n.s.
Sex	6.25	1	6.25	5.24	<.05
Treatment * Sex	.01	2	0.00	0.00	n.s.
Error	11.92	10	1.19		

Table V-10

Correlations Between Math and Strategy Measures

	CTBS Math Concepts	Math Pre	Math Post	Strategy Post
CTBS Math Computation	0.646	0.204	0.668	0.473
CTBS Math Concepts		0.367	0.823	0.526
Math Pretest			0.415	0.547
Math Posttest				0.766

Responses to the training were mixed. Half of the students felt it was confusing or boring; half did not. Four of the six students in the Math Training group felt their training was not boring and not confusing whereas five of the seven in the Strategy and Math Group felt their training was confusing, and all but one thought it was boring. One in the Math group said it was better than his usual math class! Only 8 of 13 said they could remember it by the time they got to play the game on the computer (the next day), yet 12 out of 13 said the training was helpful. It seemed helpful to them in a variety of ways: 6 mentioned being helpful with math (e.g. negative numbers, parentheses, making big numbers), 2 mentioned it helped with strategy, 3 that it helped orient them to the game. Most (14) did not want help on anything else, but 4 did want help: 2 on practice making number sentences, 1 with negative numbers, 1 with knowing what would happen when the computer said the number sentence result was wrong.

The last question on the interview was: What 3 things about the game would you tell a friend to help him/her out? Nine people would have told their friend to try to bump the computer (their opponent)! Five more would have warned the friend to try to avoid being bumped. Clearly bumping made a big impression. Five students told their friends to try to get high numbers; only one student told her friend that high numbers weren't always best (an important distinction). Four told the friend to use the coach (i.e., ask for hints) and one specifically mentioned not to be embarrassed to ask for hints because "you'll need it and it'll help." Overall, there were 11 math-oriented comments (e.g., practice number sentences, learn to

add good and fast, practice with spinner numbers ahead of time, etc.); there were 22 strategy-oriented responses (e.g., try to get 0 on a town, try to use the shortcuts, don't get bumped, etc.); and there were 13 other responses that might be classified as pertaining to rules or general game process or behavior (e.g., look where you're going, let the computer go first, read all the coach's comments, stay calm, learn how to use the mouse). There were no noticeable group differences in these comments.

Interpretations of the Effectiveness Studies

Given the significant constraints imposed by the last minute delivery of equipment and the limited availability of subjects (it was the next to the last week of school and each student could only be excused from class for about two hours total), it was not possible to mount as comprehensive an effectiveness study as originally intended. Therefore the findings of the two WEST studies were not really surprising but were nonetheless disappointing in the lack of effects on dependent measures. In short, attempts to strengthen instructional options fell short of goals. This result may be because of an overly short treatment time, ambitious goals that require prerequisite skills unaddressed in the program, or simply ineffective instructional interventions.

Still, more than half of the students reported they learned something from playing the game, and most of these (9 out of 11) said they learned something about math. This suggests that at least some degree of learning did occur but that it was too subtle to be detected by the measures used given the relatively short intervention and small number of subjects.

Several help-related findings have implications for future design efforts. Asking for hints and enjoying the coach's interventions indicate that students want help. The more able participants asked for hints more frequently. This finding is in agreement with recent research on help-seeking (Nelson-Le Gall, 1981).

The present experience with WEST underscored the importance of sufficient time in a discovery learning environment. Forty-five minutes was clearly not long enough for elementary students to become familiar with the hardware (mouse and monitor display), the game rules and the general procedures, and then proceed to learn or practice math skills and learn game strategy skills effectively. Most students barely finished three games, which seems to have been insufficient to learn or practice math skills such as negative numbers and correct use of parentheses. Future research on WEST or WEST-like games should allow considerably more time for the subject to interact with the game.

It would probably also be advantageous to allow students an introductory period during which they could become familiar with the machine and game (including one or more practice games that "don't count") prior to any training. The training would then be less academic and more relevant to their albeit short experience with the game. Perhaps they would be less bored and more likely to learn what is presented. This suggestion is supported by the fact that a majority of students in the math training group felt that was not boring or confusing, whereas a majority of the students receiving strategy and math training did feel it was boring and confusing.

Another potentially interesting variable to investigate was suggested by the vast differences in the time students took to make their moves, ranging from seven seconds to six minutes per move.

And a final set of interesting variables concerns the questioning behavior observed. In this study boys asked more questions than girls, and there was a hint of possible treatment effects for girls only. Given Burbules and Reese's (1984) results with "Rocky's Boots," one might wonder whether boys ask different questions than girls and whether there might be a training by sex interaction regarding questioning behavior.

Concerns about instructional modification of WEST involve the issue of direct instruction as a compatible option to guided discovery. We would imagine that WEST effectiveness could be remarkably enhanced if there were direct instruction loops, with models and practice available, requiring student articulation of key concepts.

CONCLUSIONS AND RECOMMENDATIONS

The focus of this project was a formative evaluation of the WEST ICAI program involving three phases: an analysis of its theoretical orientation; a formative review of its intentions, instructional strategies, and content; and studies of the program's instructional and attitudinal effects. Given this analysis, what can be concluded about WEST and what recommendations can be made for future research, development, and implementation?

WEST has numerous favorable attributes. It presents important subject matter (math skills and game strategy) in a unique way, utilizing a game format that is motivational and enjoyable. It attempts to address individual needs based on actual student performance by providing information and feedback to facilitate student exploration. It also provides opportunity for practice, and its content and format are alterable.

The basic weakness of WEST is lack of instructional expertise. It is based on the assumed effectiveness of learning through exploration with minimal guidance. Thus it may be quite inappropriate for the many learners who lack the ability or motivation to learn from exploring without adequate assistance. WEST consists mostly of limited practice and feedback, with even more limited intervention and remediation. It assumes that the basic math skills necessary to the game have been presented elsewhere although these entry skills have not been specified. Further, there are no stated objectives (other than to learn how to play the game and practice math

skills). It does contain a dozen explicitly stated principles guiding the when and what of tutoring, but unfortunately not the how of it. The instructional strategies employed relate more specifically to ICAI characteristics than to teaching-learning processes. In addition, the coaching is general rather than specific to given math or strategy skills. Furthermore, WEST failed to be as adaptive as desired. For example, the more able students received less coaching. A good human tutor or CAI system would provide an equal amount but different content to those learners. In short, WEST, like a number of more recent ICAI efforts, has ignored empirically tested principles of instruction and has failed to address the educational use of its diagnostic capabilities. Its instructional effectiveness is yet to be fully demonstrated.

Since WEST was developed a decade ago its developers have moved on to other projects, and so future revision and development of WEST itself is unlikely. The following recommendations for future research, development, and implementation of WEST-like programs, however, should be of interest because the issues raised are still germane to current ICAI work (Sleeman & Brown, 1982).

Development Recommendations:

1. Provide complete documentation and user manuals, including specification of learner objectives, appropriate criterion measures, necessary entry skills, and suggestions for educational implementation.
2. Work on a student model that includes the trial and error behavior comprising attempts to formulate a move.

3. Include some testing within the program to verify the coach's inference about what the student is trying to do and what the student knows and does not yet know, in order to sharpen diagnosis and prescription, and reduce frustration.

4. Allow some student control of program, e.g., the difficulty level of the game, and the content and timing of some of the coaching provided.

5. Provide greater instructional guidance, not just information, with more examples and non-examples.

6. Engage the learner in articulating cognitive concepts and problem solving processes.

7. Provide richer feedback, including more specific information about how to fix mistakes (rather than simply saying the answer was wrong).

8. Revise record of student behavior that is available to instructor to incorporate variables that are most relevant to educational use of results.

9. Improve motivational and attentional aspects of visual displays via the use of variety, humor, color, sound, graphics, highlighted ideas, user control of timing, and so forth.

10. Follow a complete product development cycle including: learning and instruction research knowledge incorporated in design phase, component orientation to design phase, and use of process and outcome data from effectiveness studies in systematic revisions.

The formative review and effectiveness studies led to the following suggestions for future research with WEST or WEST-like programs.

1. Enhance learning effectiveness (including metaskills) through team play; e.g., two students might play as a team against the computer and thus be able to discuss the computer's moves, their moves, the values of various options, and ways to combine the numbers to obtain different values.

2. Enhance learning via other versions of adjunct instructional materials in which content, timing and format are varied.

3. Enhance learning by lowering the computer's playing ability or spinner numbers at first and gradually fading the handicap.

4. Investigate the effects on learning of individual differences in playing styles, e.g., the content and quantity of questions, response time, number of hints/coaching requested, and so forth.

Effective implementation of WEST and WEST-like programs in the Army or other educational settings will require instructional enhancements as mentioned above. Programs of this type are simply not sufficiently instructional to be used without substantial supplements or adaptation, especially with certain students, such as those with low motivation, low frustration thresholds, low reading ability, or high computer anxiety.

Equally important when considering implementation is the fact that developers of such ICAI programs have often failed to consider precisely how to make use of them educationally. When goals and objectives are vaguely stated, how can the program's effectiveness be measured? Even when ICAI programs contain powerful diagnostic tools, there is seldom any specification of how they are to be used in the classroom. What role should a human teacher play in conjunction with this technology? There are

already lots of attractive, innovative educational products that sit on shelves, unused for the same reasons. How can this fate be avoided for ICAI programs? This significant challenge will require the combined efforts of developers, users and educational evaluators.

WEST REPORT REFERENCES

- Anderson, R.C. & Faust, G.W., (1973). Educational psychology. New York: Harper & Row.
- Barr, A. & Feigenbaum, E. (Eds.) (1982). #C4. WEST. In Handbook of artificial intelligence, Vol. II, Los Altos, CA: William Kaufman, Inc.
- Bass, R.K. and Dill, C.R. (Eds.) (1984). Instructional development, state of the art, II, 221-242. Dubuque, Iowa: Kendall/Hunt.
- Brown, A.L., Campione, J.C. & Day, J.D. (1981) Learning to learn: In training students to learn from texts. Educational Researcher, 14-21.
- Brown, J.S. & Burton, R.R. (1978). Diagnostic models for procedural bugs in basic mathematical skills. Cognitive Science, 2, 155-192.
- Brown, J.S. Burton, R.R. & Clancy, W.J. (1984). Applications of artificial intelligence to training and education. AAAI 1984 Conference Tutorial Program, 48-56.
- Burbules, N.C. & Reese, P. (1984, April). Teaching logic to children: An exploratory study of "Rocky's Boots". Paper presented at American Educational Research Association meeting, New Orleans, LA.
- Burton, R.R. & Brown, J.S. (1982). An investigation of computer coaching for informal learning activities. In Intelligent Tutoring Systems. Sleeman, D. & Brown, J.S., (Eds.) New York: Academic Press, Inc.
- Carpenter, T.P. & Moser, J.M. (1982). The development of addition and subtraction problem-solving skills. Addition and subtraction: A cognitive perspective. New Jersey: L. Erlbaum and Associates.
- Cohen, V.B.L. (1982) Evaluating instructional software for the microcomputer. Presented at the Annual Meeting of American Educational Research Association, New York, New York.
- Dugdale, S. & Kibbey, J.D. (1977, July). Elementary mathematics with PLATO. Urbana, IL: CBE Research Laboratory.
- Evaluating instructional software. Network fact sheet. (1984, February) Rosslyn, VA: Military Educators Resource Network.
- Goldstein, I. (1979). The genetic epistemology of rule systems. International Journal of Man-Machine Studies, 11 51-77.
- Guthrie, T.J. (1967). Expository instruction vs. a discovery method. Journal of Educational Psychology, 58, 45-49.

- Hewson, P.W. & Posner, G.J. (1984). The use of schema theory in the design of instructional materials. Instructional Science, 13, 119-139.
- Laughery, R.K. (1984, June). Teaching humans game-playing skills. Simulation and Games, 15, 187-212.
- Markle, S.M. (1983). Designs for instructional designers. Champaign, IL. Stips Publishing Co..
- Merrill, M.D. (1983). Component display theory. In Reigeluth, Charles M. Instructional design theories and models. Hillsdale, N.J.: Lawrence Erlbaum & Associates.
- Merrill, D. (1984) What is learner control? In Bass, R.K. & Dill, C.R. (Eds.) Instructional development, state of the art II. Dubuque, Iowa: Kendall/Hunt.
- Montague, W.E., Ellis, J.A. & Wulfecck, W.H. (1981). After years of instructional research do we know more than grandma did about how to teach people? San Diego, CA 92152: Navy Personnel Research and Development Center.
- Nelson - Le Gall, S. (1981). Help-seeking: An understudied problem solving skill in children. Developmental Review, 1, 224-246.
- O'Neil, H.F., Jr. (Ed.) (1979). Procedures for instructional systems development. New York: Academic Press.
- Reigeluth, C.M. (1983). Instructional design theories and models. Hillsdale, N.J.: Lawrence Erlbaum Associates.
- Resnick, C.A. (1975) Computational models of learners for computer assisted learning. Unpublished doctoral dissertation, University of Illinois, Urbana.
- Seiler, B.A. & Weaver, C.S. (1976). Description of PLATO whole number arithmetic lessons. Urbana, IL: Computer-Based Education Research Laboratory, University of Illinois.
- Shymansky, J., Kyle, W., & Alport, J. (1982, Nov/Dec). How effective were hands-on science programs of yesterday? Science and Children.
- Sleeman, D. & Brown, J.S. (Eds.) (1982). Intelligent Tutoring Systems. New York: Academic Press.
- Walker, D.F. & Hess, R.D., (Eds.) (1984). Instructional software, principles and perspectives for design and use. Belmont, CA: Wadsworth Publishing Co.
- Webb, N.M. (1984). Microcomputer learning in small groups: Cognitive requirements and group processes. Journal of Educational Psychology, 76(6), 1076-1088.

- Wittrock, M.C. (1963). Verbal stimuli in concept formation: Learning by discovery. Journal of Educational Psychology, 54, 183-90.
- Worthen, B.R. (1968). Discovery and expository task presentation in elementary mathematics. Journal of Educational Psychology Monograph Supplement, 59, No. 1 Part 2.

VI. IMPLICATIONS FOR FUTURE ICAI EFFORTS

This project was designed to conduct a formative evaluation of existing models of intelligent computer-assisted instruction and to explore ways to use AI in support of instructional design. In the course of conducting the project a number of lessons were learned. These lessons fall into two general categories: lessons related to the conduct of evaluation studies and how to facilitate their management in the future; and lessons related to the instructional development cycle and how to increase the effectiveness of its products.

Management Issues

The Xerox Maintenance Tutor project experience exemplifies the communication problem and the need to promote better channels of communication and to make clear lines of authority. The communication issue, however, is part of the larger problem of assuring the participation of developers in the evaluation of their efforts and/or their compliance with evaluation requirements. While the compliance issue is most salient in the Xerox case, it is also a minor theme in the PROUST effort: participation depended on the enthusiasm of the developer and, although well intentioned, this enthusiasm did not necessarily result in timely completion of agreed upon tasks. Resulting delays impeded our ability to perform.

How might compliance be facilitated in the future? An answer lies in available incentives and supports for participation. At the present, as alluded to above, there are few, but several disincentives may exist (e.g., fear of negative results). If funders are seriously interested in evaluating the results of the projects they fund, then they may need to

build evaluation into study requirements and to take responsibility for monitoring the completion of these requirements. Clearer lines of responsibility and authority and additional oversight by funders, for example, might have helped to solve the Xerox problem.

Funding requirements are one way to facilitate compliance; peer pressure and collegiality may be a second avenue that might simultaneously encourage developers to participate and increase their confidence in the results and credibility of any evaluation findings. The ICAI developers and project evaluators were from different professional communities; the credibility of the latter was perhaps moot for Xerox developers, decreasing their willingness to participate in joint efforts. An advisory board of esteemed colleagues in the AI community perhaps would have increased the credibility, prestige and power accorded to the evaluation effort and thereby have facilitated its completion. Future evaluation studies, in short, should consider constituting an advisory board to strengthen and support their efforts.

Improving the Instructional Development Cycle

The development of the WEST and PROUST programs was intended to serve a number of purposes; program developers, for example, appeared at least equally as interested in exploring the limits of the technology as in devising an effective program which was to meet particular outcomes. As a result, the development cycle did not necessarily follow a traditional model. In addition, because developers were not from a field where learning and instruction are paramount interests, they may have neglected important research principles that have emerged in such fields. Instead, they appeared to rely on their own perceptions and in some ways "reinvent

the wheel" and did not benefit from wheels which have already been invented. Incorporating lessons learned in instructional technology and in research in learning and instruction might well have increased the instructional effectiveness of both the WEST and the PROUST programs. Therefore, to increase the efficacy of future ICAI products, it may be well to build in mechanisms to assure a more systematic development process that draws on the available knowledge from the fields of education and the psychology of learning and instruction. Among the mechanisms that might be considered are the following:

 Incentives to assure that the development process focuses on particular instructional goals;

 Supports to assure that formative evaluation is an integral part of the development process and that products are successively refined to better meet instructional goals;

 A team approach to the development process which includes persons with expertise in instruction as well as those who are expert in computer technology, user interfaces, and artificial intelligence.

 The evaluation process itself may be an important tool for encouraging these changes in the development process by making expectations clear to developers and by promoting accountability. The importance of the evaluation tool, however, will be moderated by the management issues discussed earlier. Evaluation can promote more effective products but only when evaluation is supported by procedure to assure its implementation.

APPENDIX A:
for
III. Evaluation of PROUST

Illustration of I/O Usefulness To PROUST

There are three reasons for advocating the addition of I/O analysis in a programming tutor: accuracy, relevance of feedback, and transference of skills learned. Each is illustrated here in turn.

Accuracy. In the early empirical studies at Yale it was discovered that PROUST found a number of bugs which the human checkers missed. This certainly is not surprising if all the checkers had to look at was the code itself. Often, however, a tutor will ask the student, "Well, what did the program do when you ran it?" If the student answers, "It just went off to never-never land," the tutor knows to look for an endless loop. Another strategy tutors often use is "playing computer." That is, determining what the computer will do with each of the statements in the program.

When PROUST was first examined for this evaluation it was felt that these human tricks would not be necessary for a computer-based analyzer. However, a number of inaccuracies in PROUST's output were found which could have been avoided by looking at I/O behavior. For example, the following program demonstrates a common false alarm.¹

¹ This was not a false alarm in the Yale implementation because the instructor there defined "correct input validation" in class to include an error message. However, without this additional class input, this is a false alarm.

```

program rainfall (input, output);
var
  rainfall, sumtotal, average, largestsofar : real;
  dailycounter, raincounter : integer;
begin
  readln(rainfall);
  sumtotal := 0;
  dailycounter := 0;
  largestsofar := 0;
  raincounter := 0;
  while rainfall <> 9999 do
    begin
      if rainfall >= 0 then
        begin
          dailycounter := dailycounter + 1;
          if rainfall > largestsofar then
            largestsofar := rainfall;
          if rainfall > 0 then
            begin
              raincounter := raincounter + 1;
              sumtotal := sumtotal + rainfall;
            end;
          end;
        readln(rainfall)
      average := sumtotal / dailycounter;
      writeln('In ', dailycounter : 1, 'days, there were', raincounter : 1,
        'rain
        days');
      writeln(average);
      writeln(largestsofar);
    end.

```

The feedback to the learner, after analyzing this program, included the statement:

3. This program appears to be missing an input validation.

Input validation in this assignment refers to making sure that the user does not input negative values. It appears however that the loop which begins with the statement:

```

  if rainfall >= 0 then

```

would protect the program from negative input. While it is true that the student has not yet included any way for the user to know that a negative input is not being accepted, there is input validation.

The problem is that this student has used a rather sloppy approach to the problem. Thus PROUST was apparently unable to recognize the plan. Similarly, this evaluation found the plan unorthodox, but a simple strategy for determining if it would work was employed: the program was run, giving it negative values. It would thus seem that I/O analysis would have avoided even looking for the input validation bug, or at least knowing that it was not the relevant bug after goal analysis.

Feedback relevance. Even if PROUST did not use I/O behavior in its analysis, it would be important to refer to it in its feedback. The relevance of a bug to the learner is that it prevents a program from doing what it is supposed to do. It would thus be desirable that the pedagogical expert refer to bugs in terms of how they will affect (hurt) the performance of the learner's program. Another feedback statement from the above program did try to use this approach:

1. The maximum is defined if there is no input. But line 29 outputs it anyway. You should output the maximum only when there is something to compute the maximum of. Perhaps you intended line 13 to serve this purpose. If so, there may be a bug there. See what happens when you enter this data in your program:

99999

Here's the correct output:

There were 0 valid rainfalls entered.

Although this feedback (which occurred quite often) includes a reference to how the bug will affect the performance of the program, it has a number of problems. First, it is unclear what the suggestion is really saying. It has been assumed here that the designers intended to say, "See what happens when you enter this data as the first data in your program."

But there is a deeper problem. PROUST seems to be implying that if the learner tried this with the present program, it would give output which was meaningless. Actually, if the student tried this, there would be no output because of another bug which was correctly identified:

2. 9999 on line 11 is probably a typo. It should be 99999.
The statement in question is:
WHILE RAINFALL <> 9999 DO...

In other words, because these two bugs interact in the performance of the program, taking PROUST's suggestion would be confusing to the learner. PROUST was programmed to consider the interaction as it affects the learner's goals, not how it affects the output behavior of the program. While this is important for understanding the learner's intention, it does not avoid this confusion mentioned above. In fact, single component failures are relatively easy to find and incorporate in automated teaching whereas interactions are very difficult with multiple component failures.¹

Skill transfer. A third reason for considering I/O behavior in a programming tutor is to enhance skill transfer. It might be sufficient that the programming expert help a student understand what is going wrong in a current assignment. A goal of the pedagogical expert, however, should be to teach skills which the student will be able to use in the future (when there is no access to PROUST) to enable him/her to program and debug. In the present study programs were sent to PROUST as soon as they had compiled successfully. Thus students received feedback before they had a chance to see what the program would do. Even if analysis is provided

¹ Multiple componnt failures should be of particular interest to the Army since they are characteristic of certian military situations.

immediately after compilation we would like to see the first feedback focus on what will happen when they run the program. This would include comments such as:

- o You cannot have a negative amount of rainfall; see what happens when you give your program a negative amount.
- o See what happens when you try to stop your program by inputting "99999."
- o If you run your program now, it will go to never-never land after you input the first value. Can you see why?
- o When you input 99999 as the first value, your program will fail. Find where the program is failing, and see if you can figure out why.

The pedagogical expert, when implemented, would have sufficient information from the present programming expert to provide this kind of feedback. However the present implementation of PROUST could not use this approach as it is not sufficiently directive. If the student were not able to use these focusing hints to fix the bug, there would be no chance to give more information.

IDEALIZED PROUST DIALOGUE

The following illustrates the type of desirable interaction in future PROUST implementations that would include an ongoing model of the learner.

Included in this dialogue are several attempts by a Yale student to do the RAIN program with PROUST's current output and Dr. Soloway's annotation. Following each program and analysis are the idealized pedagogical expert's replies to the student.

Ideally, the pedagogical expert would attend to the student's major bugs before looking at minor ones. This aspect of the ideal tutor was not incorporated in the dialogue below in order to present the dialogue within the flow of the actual session. For example, in the first output the tutor talks about the fact that the average could be a real number. Ideally the tutor would have asked the learner to try using data which should lead to a fractional average, then have him/her compute the average on a calculator to see the problem. This would have required waiting until the major bugs were fixed so that this relatively minor problem could actually be observed.

Actual Session with Pseudo Output Added

76* Start of log file *

* Syntactically incorrect file at 21-FEB-1985 17:40:40.92 *

* Syntactically incorrect file at 21-FEB-1985 17:42:26.86 *

* File collected at 21-FEB-1985 17:46:21.71 *

PROGRAM ASSIGN5 (INPUT,OUTPUT);

VAR VALDAY, RAINDAY, RAINFALL, TOTAL, AVE, HIGH : INTEGER;

BEGIN

VALDAY := 0;

RAINDAY := 0;

TOTAL := 0;

HIGH := 0;

WRITELN('PLEASE INPUT AMOUNT OF RAIN IN NEW HAVEN FOR A DAY
IN INTEGERS');

READLN(RAINFALL);

WHILE RAINFALL < 0 DO

BEGIN

WRITELN('YOU MUST INPUT A POSITIVE AMOUNT');

WRITELN('PLEASE INPUT A POSITIVE AMOUNT OF RAINFALL');

READLN(RAINFALL);

END;

WRITELN('IF YOU DO NOT WISH TO CONTINUE TYPE 99999');

WHILE NOT (RAINFALL = 99999) DO

BEGIN

VALDAY := VALDAY + 1;

TOTAL := RAINFALL + 1;

IF RAINFALL > 0 THEN RAINDAY := RAINDAY + 1;

IF RAINFALL > HIGH THEN HIGH := RAINFALL;

END;

AVE := TOTAL DIV VALDAY;

WRITELN('THE NUMBER OF VALID DAYS IS ', VALDAY);

WRITELN('THE AVERAGE RAINFALL IS', AVE);

WRITELN('THE NUMBER OF RAINY DAYS IS', RAINDAY);

WRITELN('THE HIGHEST DAILY RAINFALL IS', HIGH);

END.

* Proust's output follows *

Starting bug analysis, please wait...

#####Annotation

[The following bug report is correct. Note that Proust couldn't understand the sum computation, although it could localize the faulty code that computes the sum.]

#####Annotation

NOW BEGINNING BUG REPORT:

(TO CONTINUE, PLEASE PRESS CARRIAGE RETURN)

>>> Now Reporting CRITICAL Bugs in the SETUP part of your program:

You have a separate loop for testing input validation at line 12. Once you get into the main loop, you don't test the input any more.

The statement in question is:

WHILE RAINFALL < 0 DO ...

#####Annotation

[The student only reads data outside the main processing loop. However, a read and check for valid data is needed inside that processing loop. This bug is directly related to the CRITICAL bug pointed out below in CONTROL part of the program. Unfortunately, PROUST does not explicitly draw out the relationship between these two bugs. Moreover, it appears that the above error message simply is not sufficiently informative to effectively aid students.]

#####Annotation

#####Annotation

[Because of system problems, we do NOT know if the student did press carriage return successively to see all the bugs. Next time we run PROUST in the classroom, we will attempt to collect this type of data]

#####Annotation

(TO CONTINUE, PLEASE PRESS CARRIAGE RETURN)

>>> Now Reporting MINOR Bugs in the SETUP part of your program:

The declaration for AVE on line 2 should be REAL instead of INTEGER.

That way you can get an exact value when you compute AVE.

Be sure that you use '/' when you divide.

(TO CONTINUE, PLEASE PRESS CARRIAGE RETURN)

>>> Now Reporting CRITICAL Bugs in the CONTROL part of your program:

Your main loop is missing a READ statement. As it stands your loop will process the same input value over and over.

(TO CONTINUE, PLEASE PRESS CARRIAGE RETURN)

>>> Now Reporting MINOR Bugs in the OUTPUT part of your program:

The average is undefined if there is no input. But line 31 outputs it anyway. You should output the average only when there is something to compute the average of.

(TO CONTINUE, PLEASE PRESS CARRIAGE RETURN)

>>> Now Reporting MINOR Bugs in the MISCELLANEOUS part of your program:

This analysis is incomplete. There are parts of this program that I could not understand, so this analysis may not be totally correct.

This program does not implement the sum.

I had problems with some of the code.

The statement in question is:

TOTAL := RAINFALL + 1

BUG REPORT NOW COMPLETE.

Pseudo Output----

There are some problems with your program. If it were run it would get into an endless loop. The problem is in the "WHILE NOT (RAINFALL = 99999)" loop. Once you get into this loop, RAINFALL never changes, so the condition of RAINFALL equaling 99999 can never be met.

Since this program is supposed to read in values until the user is done, there should be a loop somewhere to do this. The only input statement you have used is not in a loop, "READLN(RAINFALL)." Did you mean for this statement to read in all of the values?

>NO

Then I will assume that that statement was intended to get your first input before going into an input loop. I am still unable to find an input statement that will be repeated. This either means that you are missing an input loop completely, or you forgot to put an input statement into an existing loop. Which of these problems exists depends on what you were trying to do. Make whichever correction is appropriate for your original plan.

While you are at it, I notice a couple of other problems which you may wish to deal with.

Your program will always output the average rainfall as an integer. Even though the problem stated that the inputs should be integers, the average of a group of integers could easily be a real number. Remember that DIV only yields integer results.

I do not understand your statement "TOTAL := RAINFALL + 1." I can tell from how you compute AVE that TOTAL should store the total rainfall. What your program does now is store the current value of RAINFALL plus 1, each time through.

```
* File collected at 21-FEB-1985 17:55:00.78 *
PROGRAM ASSIGN5 (INPUT,OUTPUT);
VAR VALDAY, RAINDAY, RAINFALL, TOTAL, HIGH : INTEGER;
    ave : real;

BEGIN
    VALDAY := 0;
    RAINDAY := 0;
    TOTAL := 0;
    HIGH := 0;

    WRITELN('PLEASE INPUT AMOUNT OF RAIN IN NEW HAVEN FOR A DAY IN
INTEGERS');
    READLN(RAINFALL);
    WHILE RAINFALL < 0 DO
        BEGIN
            WRITELN('YOU MUST INPUT A POSITIVE AMOUNT');
            WRITELN('PLEASE INPUT A POSITIVE AMOUNT OF RAINFALL');
            READLN(RAINFALL);
        END;

    WRITELN('IF YOU DO NOT WISH TO CONTINUE TYPE 99999');
    WHILE NOT (RAINFALL = 99999) DO
        BEGIN
            VALDAY := VALDAY + 1;
            TOTAL := RAINFALL + total;
            IF RAINFALL > 0 THEN RAINDAY := RAINDAY + 1;
            IF RAINFALL > HIGH THEN HIGH := RAINFALL;
            writeLn('please input amount of rain in New Haven for a day
in integers');
            readln(rainfall);
        END;

    AVE := TOTAL / VALDAY;

    WRITELN('THE NUMBER OF VALID DAYS IS ',VALDAY);
    WRITELN('THE AVERAGE RAINFALL IS', AVE);
    WRITELN('THE NUMBER OF RAINY DAYS IS', RAINDAY);
    WRITELN('THE HIGHEST DAILY RAINFALL IS', HIGH);

END.
* PROUST's output follows *
```

####Annotation

[Changes made by the student can be seen in lower case letters. The student corrected the running total update (TOTAL := RAINFALL + total), which PROUST pointed to as potentially being a problem. He also changed DIV to / in the average update.]

####Annotation

Starting bug analysis, please wait...
NOW BEGINNING BUG REPORT:

(TO CONTINUE, PLEASE PRESS CARRIAGE RETURN)

>>> Now Reporting CRITICAL Bugs in the SETUP part of your program:

You have a separate loop for testing input validation at line 13. Once you get into the main loop, you don't test the input any more.
The statement in question is:
 WHILE RAINFALL < 0 DO ...

###Annotation

[This transcript illustrates an interesting and important problem in bug analysis. PROUST is always guessing what the student's intentions are, and bugs are result of that analysis. Thus, if PROUST guesses incorrectly, then the bug report is inappropriate. In this particular case, we see that PROUST's analysis, as printed above is incorrect. However, it is incorrect only in the context of the change made by the student! That is, in the student's first program, listed above, it appeared that the student was reading and validating the data in the first loop, and then passing it to the second loop for additional processing. However, the change made by the student in the the above program indicates that student did not have that intention at all. Rather, by adding the readln inside the second loop, and leaving the first loop alone, it now appears that the student intended only to read and validate the first data value in the first loop. Thus, the above bug analysis is incorrect for this version of the program.]

###Annotation

(TO CONTINUE, PLEASE PRESS CARRIAGE RETURN)

>>> Now Reporting CRITICAL Bugs in the OUTPUT part of your program:

You need a test to check that at least one valid data point has been input before line 31 is executed. The average will bomb when there is no input.

###Annotation

[Now that the student corrected one bug (fixing the running total update), this allows PROUST to identify yet another critical bug.]

###Annotation

(TO CONTINUE, PLEASE PRESS CARRIAGE RETURN)

>>> Now Reporting MINOR Bugs in the OUTPUT part of your program:

The maximum is undefined if there is no input. But line 36 outputs it anyway. You should output the maximum only when there is something to compute the maximum of.

The average is undefined if there is no input. But line 34 outputs it anyway. You should output the average only when there is something to compute the average of.

BUG REPORT NOW COMPLETE.

Pseudo Output----

Now I see that you did intend to use your second loop for your input. Placing the READLN statement at the end of this loop will fix the input problems I had noted before. But now I see that you have another problem. You check to make sure that the user does not give a negative value for the first input. However if I enter a negative number anywhere after the first input, the program will accept it as valid.

I see that you changed AVE to reflect real numbers when appropriate. There is still a problem with AVE. If I enter 99999 as my first value, then VALDAY equals zero, and computing AVE will give you a division by zero error.

Even if the program did not bomb when there were no valid days, average would be meaningless, and thus should not be printed out. In fact if there were no valid days entered, then HIGH and RAINDAY are also meaningless and should not be output.

I like the way you corrected the statement which updates the value of TOTAL.

* Syntactically incorrect file at 21-FEB-1985 18:05:57.80 *

* File collected at 21-FEB-1985 18:07:34.95 *

PROGRAM ASSIGN5 (INPUT,OUTPUT);

VAR VALDAY, RAINDAY, RAINFALL, TOTAL, HIGH : INTEGER;
ave : real;

BEGIN

VALDAY := 0;
RAINDAY := 0;
TOTAL := 0;
HIGH := 0;

WRITELN('PLEASE INPUT AMOUNT OF RAIN IN NEW HAVEN FOR A DAY IN
INTEGERS');

```

READLN(RAINFALL);
  WHILE RAINFALL < 0 DO
  BEGIN
    WRITELN('YOU MUST INPUT A POSITIVE AMOUNT');
    WRITELN('PLEASE INPUT A POSITIVE AMOUNT OF RAINFALL');
    READLN(RAINFALL);
  END;

  WRITELN('IF YOU DO NOT WISH TO CONTINUE TYPE 99999');
  WHILE NOT (RAINFALL = 99999) DO
  BEGIN
    VALDAY := VALDAY + 1;
    TOTAL := RAINFALL + total;
    IF RAINFALL > 0 THEN RAINDAY := RAINDAY + 1;
    IF RAINFALL > HIGH THEN HIGH := RAINFALL;
    writeln('please input amount of rain in New Haven for a day in
integers');
    readln(rainfall);
    while rainfall < 0 do
    begin
      writeln('you must input a positive amount of rainfall');
      writeln('Please input a positive amount of rainfall');
      readln(rainfall);
    end;
  END;

  AVE := TOTAL / VALDAY;

  WRITELN('THE NUMBER OF VALID DAYS IS ',VALDAY);

  IF AVE > 0 THEN WRITELN('THE AVERAGE RAINFALL IS',AVE)
  ELSE WRITELN('THERE IS NO AVERAGE RAINFALL BECAUSE IT DIDNOT
RAIN');

  WRITELN('THE NUMBER OF RAINY DAYS IS', RAINDAY);

  IF HIGH > 0 THEN WRITELN('THE HIGHEST DAILY RAINFALL IS',HIGH)
  ELSE WRITELN('THERE IS NO HIGHEST DAILY RAINFALL');

```

END.

* PROUST's output follows *

####Annotation

[The student fixed the input validation in the second loop (the loop beginning while rainfall < 0...). The student also tried to fix the bugs identified by PROUST previously in the OUTPUT section of the program. Unfortunately, the student's patches are incorrect.] the tests as boundary condition tests. Thus PROUST again reports these bugs. It would be better to tell the student how he failed to address the bugs -- since he clearly did attempt to fix them. However, that type of interaction is not in PROUST's scope.

####Annotation

Starting bug analysis, please wait...

NOW BEGINNING BUG REPORT:

(TO CONTINUE, PLEASE PRESS CARRIAGE RETURN)

>>> Now Reporting CRITICAL Bugs in the OUTPUT part of your program:

You need a test to check that at least one valid data point has been input before line 37 is executed. The average will bomb when there is no input.

(TO CONTINUE, PLEASE PRESS CARRIAGE RETURN)

>>> Now Reporting MINOR Bugs in the OUTPUT part of your program:

The maximum is undefined if there is no input. But line 46 outputs it anyway. You should output the maximum only when there is something to compute the maximum of.

The average is undefined if there is no input. But line 41 outputs it anyway. You should output the average only when there is something to compute the average of.

BUG REPORT NOW COMPLETE.

Pseudo Output-----

Your program will now check for negative inputs every time.

I see that you tried to avoid outputting the average when it is inappropriate. But I think you are checking the wrong variable. If there had been a large number of valid days, all without rainfall, the average would be zero, and should be printed out. Also, you still have no protection for division by zero when there have been no valid inputs (AVE := TOTAL / VALDAY).

But back to your output; is it appropriate to output any of the statistics if the user has not input any valid days?

File collected at 21-FEB-1985 18:11:54.09
PROGRAM ASSIGN5 (INPUT,OUTPUT);
VAR VALDAY, RAINDAY, RAINFALL, TOTAL, HIGH : INTEGER;


```

    ave : real;

BEGIN
    VALDAY := 0;
    RAINDAY := 0;
    TOTAL := 0;
    HIGH := 0;

    WRITELN('PLEASE INPUT AMOUNT OF RAIN IN NEW HAVEN FOR A DAY IN
    INTEGERS');
    READLN(RAINFALL);
    WHILE RAINFALL < 0 DO
        BEGIN
            WRITELN('YOU MUST INPUT A POSITIVE AMOUNT');
            WRITELN('PLEASE INPUT A POSITIVE AMOUNT OF RAINFALL');
            READLN(RAINFALL);
        END;

    WRITELN('IF YOU DO NOT WISH TO CONTINUE TYPE 99999');
    WHILE NOT (RAINFALL = 99999) DO
        BEGIN
            VALDAY := VALDAY + 1;
            TOTAL := RAINFALL + total;
            IF RAINFALL > 0 THEN RAINDAY := RAINDAY + 1;
            IF RAINFALL > HIGH THEN HIGH := RAINFALL;
            writeln('please input amount of rain in New Haven for a day in
            integers');
            readln(rainfall);
            while rainfall < 0 do
                begin
                    writeln('you must input a positive amount of rainfall');
                    writeln('Please input a positive amount of rainfall');
                    readln(rainfall);
                end;
            END;

            IF VALDAY > 0 THEN AVE := TOTAL / VALDAY;

            WRITELN('THE NUMBER OF VALID DAYS IS ',VALDAY);

            IF AVE > 0 THEN WRITELN('THE AVERAGE RAINFALL IS',AVE)
            ELSE WRITELN('THERE IS NO AVERAGE RAINFALL BECAUSE IT DIDNOT
            RAIN');

            WRITELN('THE NUMBER OF RAINY DAYS IS', RAINDAY);

            IF HIGH > 0 THEN WRITELN('THE HIGHEST DAILY RAINFALL IS',HIGH)
            ELSE WRITELN('THERE IS NO HIGHEST DAILY RAINFALL');

        END.
    * PROUST's output follows *

```

####Annotation

[This time the student did manage to guard the average calculation (from a potential division by zero bug). However, the student still failed to fix the other output bugs correctly pointed out by PROUST.]

####Annotation

Starting bug analysis,

NOW BEGINNING BUG REPORT:

(TO CONTINUE, PLEASE PRESS CARRIAGE RETURN)

>>> Now Reporting MINOR Bugs in the OUTPUT part of your program:

The maximum is undefined if there is no input. But line 46 outputs it anyway. You should output the maximum only when there is something to compute the maximum of.

The average is undefined if there is no input. But line 41 outputs it anyway. You should output the average only when there is something to compute the average of.

BUG REPORT NOW COMPLETE.

Pseudo Output----

Very good! You have now adequately protected against dividing by zero when there has been no valid input.

However, you are still giving ouput when there has been no valid input.

####Annotation

[At this point the student leaves the terminal. The program still has a few minor bugs. The student returns two days later, as indicated in the header information, to attempt to fix the remaining bugs.]

####Annotation

* Syntactically incorrect file at 23-FEB-1985 12:30:09.30 *

* File collected at 23-FEB-1985 12:31:38.55 *

PROGRAM ASSIGN5 (INPUT,OUTPUT);

VAR VALDAY, RAINDAY, RAINFALL, TOTAL, HIGH : INTEGER;

ave : real;

```

BEGIN
  VALDAY := 0;
  RAINDAY := 0;
  TOTAL := 0;
  HIGH := 0;

  WRITELN('PLEASE INPUT AMOUNT OF RAIN IN NEW HAVEN FOR A DAY IN
INTEGERS');
  READLN(RAINFALL);
  WHILE RAINFALL < 0 DO
    BEGIN
      WRITELN('YOU MUST INPUT A POSITIVE AMOUNT');
      WRITELN('PLEASE INPUT A POSITIVE AMOUNT OF RAINFALL');
      READLN(RAINFALL);
    END;

  WRITELN('IF YOU DO NOT WISH TO CONTINUE TYPE 99999');
  WHILE NOT (RAINFALL = 99999) DO
    BEGIN
      VALDAY := VALDAY + 1;
      TOTAL := RAINFALL + total;
      IF RAINFALL > 0 THEN RAINDAY := RAINDAY + 1;
      IF RAINFALL > HIGH THEN HIGH := RAINFALL;
      writeln('please input amount of rain in New Haven for a day in
integers');
      writeln('if you do not wish to continue type 99999');
      readln(rainfall);
      while rainfall < 0 do
        begin
          writeln('you must input a positive amount of rainfall');
          writeln('Please input a positive amount of rainfall');
          readln(rainfall);
        end;
    END;

    IF VALDAY > 0 THEN AVE := TOTAL / VALDAY;

    WRITELN('THE NUMBER OF VALID DAYS IS',VALDAY);

    IF AVE > 0 THEN WRITELN('THE AVERAGE RAINFALL IS',AVE:8:2);

    WRITELN('THE NUMBER OF RAINY DAYS IS', RAINDAY);

    IF HIGH > 0 THEN WRITELN('THE HIGHEST DAILY RAINFALL IS',HIGH);

  END.
  * PROUST's output follows *

```

####Annotation

[Unfortunately, the student still can't seem to fix the bugs. This type of situation cries out for more interaction on the part of a computer-based tutor; the system we are currently working on, BERTIE, should provide the student with assistance in this type of painful situation.]

####Annotation

:

Starting bug analysis, please wait...

NOW BEGINNING BUG REPORT:

(TO CONTINUE, PLEASE PRESS CARRIAGE RETURN)

>>> Now Reporting MINOR Bugs in the OUTPUT part of your program:

The maximum is undefined if there is no input. But line 46 outputs it anyway. You should output the maximum only when there is something to compute the maximum of.

The average is undefined if there is no input. But line 42 outputs it anyway. You should output the average only when there is something to compute the average of.

BUG REPORT NOW COMPLETE.

Pseudo Output----

I am not sure what you were trying to change. The problem still remains; your program should probably not supply any output when the user has not supplied any valid input.

:

* File collected at 23-FEB-1985 14:42:09.37 *

PROGRAM ASSIGN5 (INPUT,OUTPUT);

VAR VALDAY, RAINDAY, RAINFALL, TOTAL, HIGH : INTEGER;

ave : real;

BEGIN

VALDAY := 0;

RAINDAY := 0;

TOTAL := 0;

HIGH := 0;

```

WRITELN('PLEASE INPUT AMOUNT OF RAIN IN NEW HAVEN FOR A DAY IN
INTEGERS');
READLN(RAINFALL);
WHILE RAINFALL < 0 DO
BEGIN
    WRITELN('YOU MUST INPUT A POSITIVE AMOUNT');
    WRITELN('PLEASE INPUT A POSITIVE AMOUNT OF RAINFALL');
    READLN(RAINFALL);
END;

WHILE NOT (RAINFALL = 99999) DO
BEGIN
    VALDAY := VALDAY + 1;
    TOTAL := RAINFALL + total;
    IF RAINFALL > 0 THEN RAINDAY := RAINDAY + 1;
    IF RAINFALL > HIGH THEN HIGH := RAINFALL;
    writeln('please input amount of rain in New Haven for a day in
    integers');
    writeln('if you do not wish to continue type 99999');
    readln(rainfall);
    while rainfall < 0 do
    begin
        writeln('you must input a positive amount of rainfall');
        writeln('Please input a positive amount of rainfall');
        readln(rainfall);
    end;
END;

IF VALDAY > 0 THEN AVE := TOTAL / VALDAY;

WRITELN('THE NUMBER OF VALID DAYS IS ',VALDAY);

IF AVE > 0 THEN WRITELN('THE AVERAGE RAINFALL IS ',AVE:8:2);

WRITELN('THE NUMBER OF RAINY DAYS IS', RAINDAY);

IF HIGH > 0 THEN WRITELN('THE HIGHEST DAILY RAINFALL IS',HIGH);

END.
* PROUST's output follows *

####Annotation
[While the student is making changes to his program, these changes are
getting fixed the problem.]
####Annotation

:
Starting bug analysis, please wait...

```

NOW BEGINNING BUG REPORT:

(TO CONTINUE, PLEASE PRESS CARRIAGE RETURN)

>>> Now Reporting MINOR Bugs in the OUTPUT part of your program:

The maximum is undefined if there is no input. But line 45 outputs it anyway. You should output the maximum only when there is something to compute the maximum of.

The average is undefined if there is no input. But line 41 outputs it anyway. You should output the average only when there is something to compute the average of.

BUG REPORT NOW COMPLETE.

Pseudo Output----

I see that you are bewildered. Your changes have nothing to do with the problem that I noted; there should be no output when there has been no input. Remember that you have a variable which contains the number of valid days which have been input. When this is zero, then there has been no valid input.

####Annotation

[At this point the student left the terminal. The program still has some minor bugs.]

####Annotation

:

* File collected at 26-FEB-1985 19:13:52.12 *

PROGRAM ASSIGN5 (INPUT,OUTPUT);

VAR VALDAY, RAINDAY, RAINFALL, TOTAL, HIGH : INTEGER;
ave : real;

BEGIN

VALDAY := 0;
RAINDAY := 0;
TOTAL := 0;
HIGH := 0;

WRITELN('PLEASE INPUT AMOUNT OF RAIN IN NEW HAVEN FOR A DAY IN
INTEGERS');

READLN(RAINFALL);

```

WHILE RAINFALL < 0 DO
BEGIN
    WRITELN('YOU MUST INPUT A POSITIVE AMOUNT');
    WRITELN('PLEASE INPUT A POSITIVE AMOUNT OF RAINFALL');
    READLN(RAINFALL);
END;

WHILE NOT (RAINFALL = 99999) DO
BEGIN
    VALDAY := VALDAY + 1;
    TOTAL := RAINFALL + total;
    IF RAINFALL > 0 THEN RAINDAY := RAINDAY + 1;
    IF RAINFALL > HIGH THEN HIGH := RAINFALL;
    writeln('please input amount of rain in New Haven for a day in
    integers');
    writeln('if you do not wish to continue type 99999');
    readln(rainfall);
    while rainfall < 0 do
    begin
        writeln('you must input a positive amount of rainfall');
        writeln('Please input a positive amount of rainfall');
        readln(rainfall);
    end;
END;
    if rainfall = 99999 then
    begin
        IF VALDAY > 0 THEN AVE := TOTAL / VALDAY;

        WRITELN('THE NUMBER OF VALID DAYS IS ',VALDAY);

        IF AVE > 0 THEN WRITELN('THE AVERAGE RAINFALL IS ',AVE:8:2);

        WRITELN('THE NUMBER OF RAINY DAYS IS', RAINDAY);

        IF HIGH > 0 THEN WRITELN('THE HIGHEST DAILY RAINFALL IS',HIGH);
    end;
END.
* PROUST's output follows *

####Annotation
[The student is trying again to fix those last bugs. However, he actually
now introduces a new bug!]
####Annotation

:
Starting bug analysis, please wait...

NOW BEGINNING BUG REPORT:

```

(TO CONTINUE, PLEASE PRESS CARRIAGE RETURN)

>>> Now Reporting MINOR Bugs in the OUTPUT part of your program:

The maximum is undefined if there is no input. But line 46 outputs it anyway. You should output the maximum only when there is something to compute the maximum of.

The average is undefined if there is no input. But line 42 outputs it anyway. You should output the average only when there is something to compute the average of.

(TO CONTINUE, PLEASE PRESS CARRIAGE RETURN)

>>> Now Reporting MINOR Bugs in the MISCELLANEOUS part of your program:

The guard at line 36 appears to be unnecessary.
The statement in question is:
 IF RAINFALL = 99999 THEN ...

BUG REPORT NOW COMPLETE.

Pseudo Output----

Good try, but wrong. I like the way you put the whole output section under one if-statement. However the if-statement checks the wrong variable. Checking to see if the variable RAINFALL = 99999 just indicates that the last value entered was the terminator, which would always be the case. The value of VALDAY would indicate whether there were any valid days input.

Once you correct this, by the way, some of your other if-statements in the ouput section would probably be superfluous.

* File collected at 26-FEB-1985 19:24:37.28 *
PROGRAM ASSIGN5 (INPUT,OUTPUT);
VAR VALDAY, RAINDAY, RAINFALL, TOTAL, HIGH : INTEGER;
 ave : real;

BEGIN
 VALDAY := 0;
 RAINDAY := 0;
 TOTAL := 0;
 HIGH := 0;

 WRITELN('PLEASE INPUT AMOUNT OF RAIN IN NEW HAVEN FOR A DAY IN
INTEGERS');
 READLN(RAINFALL);


```

WHILE RAINFALL < 0 DO
BEGIN
    WRITELN('YOU MUST INPUT A POSITIVE AMOUNT');
    WRITELN('PLEASE INPUT A POSITIVE AMOUNT OF RAINFALL');
    READLN(RAINFALL);
END;

WHILE NOT (RAINFALL = 99999) DO
BEGIN
    VALDAY := VALDAY + 1;
    TOTAL := RAINFALL + total;
    IF RAINFALL > 0 THEN RAINDAY := RAINDAY + 1;
    IF RAINFALL > HIGH THEN HIGH := RAINFALL;
    writeln('please input amount of rain in New Haven for a day in
    integers');
    writeln('if you do not wish to continue type 99999');
    readln(rainfall);
    while rainfall < 0 do
    begin
        writeln('you must input a positive amount of rainfall');
        writeln('Please input a positive amount of rainfall');
        readln(rainfall);
    end;
END;

IF VALDAY > 0 THEN
begin
    ave := total/valday;
    writeln('the number of valid days is',valday);
    writeln('the average rainfall is', ave:8:2);
    if high > 0 then writeln('the highest daily rainfall is',high);
end;
END.
* PROUST's output follows *

####Annotation
[In the intervening 11 minutes, the student apparently sought and received
help, since in the above version he has radically transformed his program.
There is still one bug left; this appears, however, to have been an
oversight.]
####Annotation

:
Starting bug analysis, please wait...

NOW BEGINNING BUG REPORT:

(TO CONTINUE, PLEASE PRESS CARRIAGE RETURN)

```

>>> Now Reporting CRITICAL Bugs in the OUTPUT part of your program:

There is no output statement for the rainy day counter.

BUG REPORT NOW COMPLETE.

Pseudo Output-----

Excellent, you have corrected all of the problems which I have seen. The only problem I now see is that for some reason you took out your old statement for writing the number of rainy days. If you put that back in, I think you will have it!

```
-----
* File collected at 26-FEB-1985 19:27:41.73 *
PROGRAM ASSIGN5 (INPUT,OUTPUT);
VAR VALDAY, RAINDAY, RAINFALL, TOTAL, HIGH : INTEGER;
    ave : real;

BEGIN
    VALDAY := 0;
    RAINDAY := 0;
    TOTAL := 0;
    HIGH := 0;

    WRITELN('PLEASE INPUT AMOUNT OF RAIN IN NEW HAVEN FOR A DAY IN
INTEGERS');
    READLN(RAINFALL);
    WHILE RAINFALL < 0 DO
        BEGIN
            WRITELN('YOU MUST INPUT A POSITIVE AMOUNT');
            WRITELN('PLEASE INPUT A POSITIVE AMOUNT OF RAINFALL');
            READLN(RAINFALL);
        END;

    WHILE NOT (RAINFALL = 99999) DO
        BEGIN
            VALDAY := VALDAY + 1;
            TOTAL := RAINFALL + total;
            IF RAINFALL > 0 THEN RAINDAY := RAINDAY + 1;
            IF RAINFALL > HIGH THEN HIGH := RAINFALL;
            writeln('please input amount of rain in New Haven for a day in
            integers');
            writeln('if you do not wish to continue type 99999');
            readln(rainfall);
            while rainfall < 0 do
                begin
                    writeln('you must input a positive amount of rainfall');
                    writeln('Please input a positive amount of rainfall');
                    readln(rainfall);
                end;
        END;
```

```

END;

IF VALDAY > 0 THEN
begin
  ave := total/valday;
  writeln('the number of valid days is',valday);
  writeln('the average rainfall is', ave:8:2);
  if high > 0 then writeln('the highest daily rainfall is',high);
  writeln('the number of rainy days is',rainday);
end;
END.
* PROUST's output follows *

```

```

####Annotation
[Finally, the student has developed a correct program.]
####Annotation

```

```

:
Starting bug analysis, please wait...

```

```

NOW BEGINNING BUG REPORT:

```

```

BUG REPORT NOW COMPLETE.

```

```

Pseudo Output----

```

```

Excellent, you have written a correct program.

```

```

-----

```

In summary, PROUST is an impressive initial product of artificial intelligence research. Its ability to recognize a wide range of user input and to identify accurately problems will be a valuable component for tutoring. Efforts to identify the steps the learner used, go far in supplying what is needed in a student model. With the addition of the pedagogical expert, there is indeed potential for a powerful tutoring system.

Some work remains with the present module. The addition of a second program-problem to its repertoire was apparently not without problems. It would be hoped, however, that each expansion leads to generalizations of the knowledge-base, making future expansions easier. Also, there has been no empirical validation of PROUST's accuracy in diagnosing the learner's intentions. While its accuracy in identifying bugs might imply that it has accurately identified intentions, the accuracy of the diagnosis should be independently verified.

As a final note, the question arises about whether the present PROUST could be recommended for use in a Pascal class. The answer is a definite, "Yes." No other existing computer program can provide the kind of feedback, of which PROUST is capable. Student access to this program at any time of day or night is a beneficial learning aid.

STUDENT INFORMATION FORM

STUDENT _____ SEX: M F SECTION NUMBER _____
IDENTIFICATION

A. BACKGROUND INFORMATION

College Status (Circle One): Freshman Sophomore Junior Senior

Major _____

SAT Scores

Math _____

English _____

Grade Point Average (Use 4.0 Scale)

College (to date) _____

High School _____

Math Courses Completed:

College-Level (Mark all that apply):

Algebra _____

Geometry _____

Trigonometry _____

Calculus _____

Other _____

Which? _____

High School-Level (Mark all that apply):

Algebra _____

Geometry _____

Trigonometry _____

Calculus _____

Other _____

Which? _____

B. COMPUTER BACKGROUND

Please circle YES or NO to the following questions,

1. Do you own a computer? YES NO
2. Do you have access to a computer other than on campus? YES NO
3. Do you know how to program in a computer language other than Pascal? YES NO
If yes, which? _____

4. How many computer-related courses have you completed, if any? (E.g., programming, data processing, etc.) _____ Number of courses completed.
5. In how many computer-related courses, if any, are you currently enrolled in addition to this course? _____ Number of current courses.
6. To how many computer-oriented magazines, if any, do you subscribe? _____ Number of subscriptions.

Please circle the letter indicating your response to the next questions.

7. How easy is it for you to use the computer editing functions, such as inserting, deleting, or rearranging numbers or text?
- A Not applicable (I have never used a computer before.)
B Very difficult
C Difficult
D Easy
E Very easy
8. Do you touch type? ("Touch typing" refers to the ability to type without looking at the keys.)
- YES NO
9. How fast do you type?
- A Don't know
B Less than 40 words per minute
C Between 40 and 55 words per minute
D Over 55 words per minute
10. How often do you use a computer for word processing, computation, data analysis, graphics, or other purposes? (Do not consider games.)
- A At least five days per week
B At least once a week
C A few times a month
D Once a month or less
E Never
11. How often do you play computer games?
- A At least five days per week
B At least once a week
C A few times a month
D Once a month or less
E Never

Please circle the number indicating your response.

12. How well do you feel that you understand computer programming?

1	2	3	4	5
Not at all		Moderately well		Very well

13. How well do you feel that you understand how computers are used to process information?

1	2	3	4	5
Not at all		Moderately well		Very well

14. In your opinion, how important is the development of computer technology?

1	2	3	4	5
Extremely unimportant		Not sure		Extremely important

15. What effect do you feel that computer technology will have by the end of this century?

1	2	3	4	5
Strong negative effect		No effect		Strong positive effect

Other comments: _____

----- C. PERSONAL OUTLOOK INVENTORY ¹ -----

Listed below are statements which allow you to express your interests and attitudes on a number of topics. None of these statements can in any way be described as representing anything good or bad. Please indicate how well each statement describes what you typically do or how you typically feel.

	STRONGLY AGREE	AGREE	DISAGREE	STRONGLY DISAGREE
1. I prefer fishing to tennis.	1	2	3	4
2. When I hear of, or read about, a new idea that sounds interesting, I typically try to think about how I can use it.	1	2	3	4
3. I think it's important to find out about my instructor's opinions before telling about my own opinions in his (her) class.	1	2	3	4
4. It would be accurate to say that I really enjoy toying with ideas.	1	2	3	4
5. I prefer to have my grade in a course based mostly on a term paper rather than multiple choice tests.	1	2	3	4
6. I would rather watch a heated debate on a controversial topic than a popular music program.	1	2	3	4
7. I would have more fun joining in a good debate than going fishing.	1	2	3	4

¹ Intellectual Self-Confidence Inventory

	STRONGLY AGREE	AGREE	DISAGREE	STRONGLY DISAGREE
8. When it matters to me, I can usually figure out how to win an argument.	1	2	3	4
9. I would rather win an argument because my <u>style</u> of speaking (voice quality, word choice, etc.) was skillful, than because my arguments were logical.	1	2	3	4
10. If given a choice, I would take a course in the use of logic rather than a course in the history of sports.	1	2	3	4
11. I think that courses in mathematics are basically a waste of limited time.	1	2	3	4
12. Most courses in the sciences, like physics and chemistry, are easy enough if you take the time to study.	1	2	3	4
13. Most teachers use uncommon technical terms just to make their classes appear difficult and their speech impressive rather than to help students to understand.	1	2	3	4
14. I think I have as much mental ability as most of my teachers.	1	2	3	4
15. I don't like teachers who make you guess what their opinion is; I think the teacher should spell out exactly what he believes.	1	2	3	4
16. I get annoyed when people use words I don't know.	1	2	3	4
17. If I hear or read a new word, and can't seem to get its meaning, then I make a point of finding out what it means.	1	2	3	4
18. I think it's best to rely on advice from my parents, or friends when I'm not sure of the right thing to do.	1	2	3	4

	STRONGLY AGREE	AGREE	DISAGREE	STRONGLY DISAGREE
19. I would rather take a guided vacation tour than take the trouble to work through the maps.	1	2	3	4
20. When I read books, I rarely check to see if the ideas presented are logically consistent.	1	2	3	4
21. I think that a collection of people working together in a group would almost always develop a better solution to a complex problem than any individual, regardless of his capability.	1	2	3	4
22. The most complicated intellectual problems are the most interesting to work out.	1	2	3	4
23. Purely mental games, like chess, usually bore me.	1	2	3	4
24. If I suspected that one of my instructors made a logical or mathematical error in working on a problem in class, I would politely point out his error.	1	2	3	4
25. Overall, entertainers, athletes, actors, and musicians have added more to mankind's happiness than scientists.	1	2	3	4
26. I would rather be a great actor, or actress, than a great scientist.	1	2	3	4
27. When I don't understand an instructor's explanation of a topic in class, I don't usually raise my hand to have him explain again.	1	2	3	4
28. If I disagreed with an instructor's answer for a multiple choice test, I would politely, but definitely, challenge the answer.	1	2	3	4

		STRONGLY AGREE	AGREE	DISAGREE	STRONGLY DISAGREE
29.	I believe that too much study is required for most school work.	1	2	3	4
30.	In general, I think most textbooks are much too hard to read.	1	2	3	4
31.	I would much rather have an instructor give me a specific assignment than have to choose my own topic.	1	2	3	4

 STUDENT IDENTIFICATION

 SECTION NUMBER

 STUDENT QUESTIONNAIRE ¹

 Please circle your answer to the following questions.

 A. PROUST OBJECTIVES

- | | | | | |
|----|---|-----|----|----------------|
| 1. | Were the purposes of PROUST presented to you on the screen? | YES | NO | DON'T REMEMBER |
| 2. | Were you told the purposes of PROUST by your instructor? | YES | NO | DON'T REMEMBER |
-

 B. LEARNER ASSESSMENT OF PROUST EFFECTIVENESS

- | | | | | | | |
|----|---|----------------|---|--------------------|---|---------------|
| 3. | How effective do you feel that PROUST was in helping you: | Very Effective | | Somewhat Effective | | Not Effective |
| | A. To use looping strategies effectively? | 1 | 2 | 3 | 4 | 5 |
| | B. To debug your Pascal programs? | 1 | 2 | 3 | 4 | 5 |
| | C. Other (Explain, _____) | 1 | 2 | 3 | 4 | 5 |
-
- | | | | |
|----|---|-----|----|
| 4. | Do you feel that PROUST has helped you to write and debug Pascal programs faster? | YES | NO |
| | Why? _____ | | |
-
- | | | | |
|----|---|-----|----|
| 5. | Would you recommend PROUST to other students? | YES | NO |
| | Why? _____ | | |
-
- | | | | |
|----|--|-----|----|
| 6. | Would you change PROUST to make it more effective? | YES | NO |
| | If yes, how? _____ | | |
| | _____ | | |
| | _____ | | |

¹ Parts A-F comprise the PROUST Questionnaire. Parts G-H are the Computer Attitude Scale.

C. LEARNER ASSESSMENT OF PROUST COMMUNICATION

7. In your opinion, how often did PROUST correctly identify your bugs on:
- | | Rarely
or never | 1 | 2 | Some-
times | 3 | Almost
always | 4 | 5 |
|--------------------------|--------------------|---|---|----------------|---|------------------|---|---|
| o The rainfall problem? | | 1 | 2 | 3 | 4 | 5 | | |
| o The checkbook problem? | | 1 | 2 | 3 | 4 | 5 | | |
8. Was the feedback from PROUST:
- | | YES | NO |
|-------------------------------|-----|----|
| A. Easy to understand? | YES | NO |
| B. Appropriate? | YES | NO |
| C. Sufficient? | YES | NO |
| D. Related to class lectures? | YES | NO |

D. AFFECTIVE FACTORS

- | | Not At
All | 1 | 2 | Some | 3 | 4 | A Great
Deal | 5 |
|--|---------------|---|---|------|---|---|-----------------|----|
| 9. How much did you enjoy using PROUST? | 1 | 2 | 3 | 4 | 5 | | | |
| 10. To what extent did PROUST motivate you to learn Pascal? | 1 | 2 | 3 | 4 | 5 | | | |
| 11. To what extent did PROUST help you feel more comfortable with: | | | | | | | | |
| A. Using the computer? | 1 | 2 | 3 | 4 | 5 | | | |
| B. Writing computer programs? | 1 | 2 | 3 | 4 | 5 | | | |
| 12. How easy was it for you: | | | | | | | | |
| A. To use the computer? | 1 | 2 | 3 | 4 | 5 | | | |
| B. To use the editor? | 1 | 2 | 3 | 4 | 5 | | | |
| C. To find the information you needed on the screen? | 1 | 2 | 3 | 4 | 5 | | | |
| 13. Would graphics have enhanced PROUST's effectiveness? | | | | | | | YES | NO |
| If yes, how? _____ | | | | | | | | |

E. STUDENT STRATEGIES--METACOGNITIVE FACTORS

The following questions ask about your strategies for completing your programming assignments.

14. Did you preplan your program on paper? YES NO
15. What was your primary approach to writing programs when using PROUST?
- A. I focused on writing a set of individual programming statements to try to accomplish the program goal. A
- B. I focused on the overall program, breaking it down into its component parts and writing a set of statements to accomplish each part. B
16. Did you collaborate with:
- A. Other students on line? YES NO
- B. Other students off line? YES NO
- C. The instructor on line? YES NO
- D. The instructor off line? YES NO

F. COMMENTS

We would appreciate any other comments you may have about PROUST.

G. COMPUTER RELATED OPINIONS

Please circle the number indicating your response.

1. How well do you feel that you understand computer programming?

1	2	3	4	5
Not at all		Moderately well		Very well

2. How well do you feel that you understand how computers are used to process information?

1	2	3	4	5
Not at all		Moderately well		Very well

3. In your opinion, how important is the development of computer technology?

1	2	3	4	5
Extremely unimportant		Not sure		Extremely important

4. What effect do you feel that computer technology will have by the end of this century?

1	2	3	4	5
Strong negative effect		No effect		Strong positive effect

5. How easy has it been for you to learn to write computer programs in PASCAL?

1	2	3	4	5
Very easy	Fairly easy	About average	Fairly hard	Very hard

6. How enjoyable have you found the work you have been doing to learn PASCAL?

1	2	3	4	5
Very frustrating		Neutral		Very enjoyable

7. How well do you feel that you can now program in PASCAL?

1	2	3	4	5
Not at all well		Moderately well		Extremely well

8. How much success do you feel that you have experienced in learning to program in PASCAL?

1	2	3	4	5
None		Some		A great deal

9. In your opinion, how valuable is PASCAL as a tool for using the computer?

1	2	3	4	5
Not at all		Some		A great deal

10. How confident would you feel if you were assigned to write a computer program in PASCAL for the kind of tasks covered in this course so far?

1	2	3	4	5
Not at all confident		Somewhat confident		Very confident

11. If you had the opportunity, how likely would you be to do additional work in PASCAL programming?

1	2	3	4	5
Very unlikely	Somewhat unlikely	Not sure	Somewhat likely	Very likely

H. COMPUTER ATTITUDE SCALE

Below are a series of statements. There are no correct answers for these statements. They are designed to permit you to indicate the extent to which you agree or disagree with the ideas expressed. Place a check mark in the parentheses under the label which is closest to your agreement or disagreement with the statement.

	Strongly agree	Slightly agree	Slightly disagree	Strongly disagree
1. Computers do not scare me at all.	()	()	()	()
2. I'm no good with computers.	()	()	()	()
3. I would like working with computers.	()	()	()	()

	Strongly agree	Slightly agree	Slightly disagree	Strongly disagree
4. Working with a computer would make me very nervous.	()	()	()	()
5. Generally I would feel OK about trying a new problem on the computer.	()	()	()	()
6. The challenge of solving problems with computers does not appeal to me.	()	()	()	()
7. I do not feel threatened when others talk about computers.	()	()	()	()
8. I don't think I would do advanced computer work.	()	()	()	()
9. I think working with computers would be enjoyable and stimulating.	()	()	()	()
10. I feel aggressive and hostile toward computers.	()	()	()	()
11. I am sure I could do work with computers.	()	()	()	()
12. Figuring out computer problems does not appeal to me.	()	()	()	()
13. It wouldn't bother me at all to take computer courses.	()	()	()	()
14. I'm not the type to do well with computers.	()	()	()	()
15. When there is a problem with a computer run that I can't immediately solve, I would stick with it until I have the answer.	()	()	()	()
16. Computers make me feel uncomfortable.	()	()	()	()
17. I am sure I could learn a computer language.	()	()	()	()
18. I don't understand how some people can spend so much time working with computers and seem to enjoy it.	()	()	()	()
19. I would feel at ease in a computer class.	()	()	()	()
20. I think using a computer would be very hard for me.	()	()	()	()

	Strongly agree	Slightly agree	Slightly disagree	Strongly disagree
21. Once I start to work with the computer, I would find it hard to stop.	()	()	()	()
22. I get a sinking feeling when I think of trying to use a computer.	()	()	()	()
23. I could get good grades in computer courses.	()	()	()	()
24. I will do as little work with a computer as possible.	()	()	()	()
25. I would feel comfortable working with a computer.	()	()	()	()
26. I do not think I could handle a computer course.	()	()	()	()
27. If a problem is left unsolved in a computer class, I would continue to think about it afterward.	()	()	()	()
28. Computers make me feel uneasy and confused.	()	()	()	()
29. I have a lot of self-confidence when it comes to working with computers.	()	()	()	()
30. I do not enjoy talking with others about computers.	()	()	()	()

CSE TOPIC ANALYSIS AND MEASUREMENT
SPECIFICATIONS: YALE EXAM

Objectives

PROUST Problems

This is a first estimate of the learning objectives which can be inferred from PROUST's present problem space. We have tried to include all objectives which an instructor might have in assigning the PROUST problems. This does not necessarily mean that they are all objectives which PROUST itself was designed to attempt.

The learner will ...

- I. be able to write looping structures
 - A. choose the appropriate looping construct
 - B. properly implement looping structures
 1. properly initialize loop parameters
 2. include appropriate termination conditions
 2. provide correct syntactic elements
- II. be able to incorporate conditionals appropriately
 - A. will choose the appropriate conditional structure
 - B. use the appropriate relational and logical operators
 - C. use appropriate test values
- III. be able to write an algorithm for averaging numerical data
 - A. include statements which will sum the data
 1. initialize variable(s) needed for summing
 2. declare variable to be of appropriate type
 3. update the sum correctly
 - B. include appropriate statements to keep track of the number of inputs
 1. initialize the counting variable
 2. declare the variable to be of type integer
 3. appropriately increment the counter
 - C. correctly compute average
 1. use appropriate variables for calculation
 2. provide protection against division by zero

- IV. be able to include the appropriate statements in a program to keep track of a maximum or minimum value
 - A. initialize variable to store max/min with appropriate value
 - B. declare variable to be of the appropriate type
 - C. use appropriate test for new min/max
 - D. use appropriate statement for replacing min/max when needed
- V. include necessary statements to provide an appropriate user interface
 - A. write statements for acquiring input
 - 1. use clear and informative prompts
 - 2. provide effective error checking for input data
 - 3. assign data input to appropriate variables
 - B. write statements to print useful output
 - 1. provide clear and appropriate labels for output data
 - 2. output all appropriate values

OBJECTIVES

PROUST PROBLEMS

This is a first estimate of the learning objectives which can be inferred from Proust's present problem space. We have tried to include all objectives which an instructor might have in assigning the Proust problems. This does not necessarily mean that they are all objectives which Proust itself was designed to attempt.

The learner will...

Levels of skill:

- ID - identify error of omission in existing program
- INC - identify error of incorrect implementation in existing program
- ART - articulat the reason something is an error in an existing program
- LOC - identify the correct location for a particular statement in a given program
- COR - generate the correct statement to accomplish a given programming task

	errors			generate	
	ID	INC	ART	LOC	COR
I. be able to write looping structures					
A. appropriate looping construct		X	X		
B. implement looping structures					
1. initialize loop parameters	X	X	X	X	X
2. termination conditions	X	X	X	X	X
3. scope of loop		X	(Y)		X
II. be able to incorporate conditionals					
A. appropriate conditional structure		X	X		
B. relational and logical operators	X	X	X	X	(U)
C. test values	X	X	X	X	X

III. be able to write an algorithm for averaging numerical data

A. statements which will sum the data

1. initialization of variable(s) needed for summing
2. declaration of variable(s)
3. updating the sum

(U)	X	X	X	X
X	X	X	X	X
X	X	Y	X	(U)

B. statements for counting

1. initialization of counting variable
- * 2. declaration of counting
3. incrementing the counter

X	X	X	X	X
X	X	X	X	X
X	X	X	X	X

C. computing average

1. calculation
2. protection against division by zero

X	X	X	X	X
(U)	X	X	X	X

IV. be able to include the appropriate statements in a program to keep track of a maximum or minimum value

A. initialization of variable to store min/max

X	X	X	X	X
---	---	---	---	---

* B. declaration of variable

X	X	X	X	X
---	---	---	---	---

C. test for new min/max

X	X	X	X	X
---	---	---	---	---

D. updating min/max

X	X	(Y)	X	X
---	---	-----	---	---

V. include necessary statements to provide an appropriate user interface

A. acquiring input

- * 1. clear and informative prompts
2. effective error checking
3. assigning data

X	X	X	X	X
X	X	X	X	X
X	X	X	X	X

B. print output

- * 1. clear and appropriate labels for output data
- * 2. output values

X	X	X	X	X
X	X	X	X	X

Sample Exam Items

Item 1: III.A.1 ID

```
1  program EXAMPLE1 (input, output);
2  const
3      STOP=9999;
4  var
5      X,SUM:integer;
6  begin
7      read (X);
8      while X<>STOP do
9          begin
10             SUM:=SUM+X;
11             read (X);
12         end;
13     writeln (SUM);
14 end.
```

Which of the following is an error in the program EXAMPLE?

- a. Missing SUM:=0; statement before line 7. *
- b. Extra read (X); statement in line 7.
- c. Missing X:=0; statement before line 7.
- d. Repeat-loop more appropriate than while-loop.
- e. None of the above.

Item 2: III.A.1 ID

```
1  program EXAMPLE2 (input,output);
2  var A,TOTAL : real;
3  begin
4      repeat
5          read(A)
6          TOTAL:=TOTAL + A;
7      until TOTAL > 200;
8      writeln( TOTAL );
9  end.
```

Which of the following is an error in the program EXAMPLE2?

- a. TOTAL is never set to zero before entering repeat-loop. *
- b. The variable A is not read before entering repeat-loop.
- c. While-loop more appropriate than repeat-loop.
- d. Variable A is never set to TOTAL.
- e. None of the above.

Item 3: III.A.1 ID

```
1  program EXAMPLE3 (input,output);
2      var
3          BOYS,GIRLS,SCORE,TOTAL : integer;
4          ANSWER                  : char;
5      begin
6          BOYS := 0;
7          GIRLS:= 0;
8          repeat
9              readln (SCORE);
10             read (ANSWER);
11             if ANSWER = 'b' then
12                 BOYS := BOYS + SCORE;
13             if ANSWER = 'g' then
14                 GIRLS:= GIRLS + SCORE;
15             until ANSWER = 'q';
16             TOTAL := BOYS + GIRLS;
17             writeln (GIRLS,BOYS,TOTAL);
18         end;
```

Which of the following is an error in the program EXAMPLE3?

- a. Missing TOTAL := 0; statement before line 8.
- b. A while-loop should be used so that the terminal SCORE is not added in.
- c. ANSWER should be read before SCORE.
- d. Lines 6 and 7 are not necessary.
- e. None of the above.

UCLA
STUDENT INTERVIEW FORM
DIRECTIONS TO INTERVIEWER

1. Use one form per student.

2. Say:

I am going to ask you a number of questions related to your use of the PROUST program.

For some of the questions, I would like you to answer YES or NO. Also, I would appreciate any additional information you can provide. I encourage you to elaborate. We want to find out as much as we can about how PROUST works so that we can improve it.

You will find that for some of the questions, you cannot give a simple YES or NO answer. In those cases, please give any information that you feel would help us understand your use of PROUST.

3. Read the questions to the student. Circle the YES or NO response. Take notes on elaborations.

STUDENT INTERVIEW FORM

Answer YES or NO where appropriate. Please elaborate whenever you have additional information.

PROUST AS AN EFFECTIVE TUTOR

1. Do you feel that PROUST was effective in teaching you to debug Pascal programs? (Probe for differences on rainfall and checkbook problems.) YES NO

Why or why not?

2. Do you feel that you will save time debugging Pascal programs because of your experience with PROUST? YES NO

Why or why not?

3. Do you feel that you will write better Pascal programs because of PROUST? YES NO

Why or why not?

4. Would you recommend PROUST to other students? YES NO

Why or why not?

5. Would you change PROUST to make it more effective in teaching Pascal?

YES NO

If yes, how?

TIME SPENT ON PROUST

6. How much time in hours and minutes do you estimate that you spent working with PROUST on the computer?
7. How much time in hours and minutes did you spend on PROUST related study without the computer?
8. Do you feel you will save time debugging your future Pascal programs now that you have used PROUST?

YES NO

Why or why not?

AFFECTIVE RESPONSE TO PROUST

9. Did you enjoy using PROUST? YES NO

Why or why not?

10. Did any feature of PROUST frustrate you? YES NO

What and why?

Other comments. We would welcome any other comments or reactions you may have to PROUST.

THANK YOU VERY MUCH

APPENDIX B
for
V. Evaluation of WEST

ICAI EVALUATION TOOL

Program Description and Background

Bibliographic Information

Title
Authors
Manufacturer/Publisher
Address
Copyright date(s)
Cost
Available for what microcomputer
 model
 memory
AI language used to write program

Components Available

Teacher's Guide
Student Guide
Program users
Other support materials

Program Application

Subject area covered
Curriculum role
 Adjunct
 Mainline
 Management only
Mode of interaction
 Drill and practice
 Tutorial
 Game
 Simulation
 Problem solving
 Exploration

Program Users

Specified target audience/intended users
Specified learner entry skills
Unintended audience

Program Intents

Developer's Rationale

Develop skills in particular area
Present concepts and principles
Supplement other resources
Apply specific instructional approach
Serve special group of learners
Introduce subject matter content in a unique or specific way
Other

Major Emphasis of Learner Objectives

Recall of previously learned facts
Application of specific skills
Abstraction beyond concrete level
Transfer of learning
Critical thinking skills
Create specific feelings and attitudes
Develop study skills
Other

Instructional Strategies

Objectives stated behaviorally
Objectives stated in terms of the learner
Objectives identify type of learning

 Type of learning

 Psychomotor

 Kinesthetic repertoires

 Chains

 Responses

 Simple Cognitive

 Associations

 Sequence learning

 Algorithms

 Verbal responses

 Complex Cognitive

 Concepts

 Principles

 Strategy learning

Objectives include higher order skills

Learners are informed of the objectives

Program Contents

Content Scope

Topics covered:

Range of content is adequate to achieve program's intents

Scope is appropriate for learning levels of intended users

Content appropriately includes:

- Facts

- Concepts

- Procedures

- Principles

Content Sequence

Sequence is specified by developer

Content is sequenced according to:

- Chronology

- Natural unit sequencing

- Problem-centered

- Task analysis

- Dictated by learning approach (i.e. game)

Sequence is set by:

- Program

- Teacher

- Student

Student's entry into and out of an activity:

- is fixed by program in set sequence and learner must

continue

- through whole sequence

- can be varied by user

- can be varied by teacher

Sequence not important because verbal information is taught

Students taught to use control for effective learning

Cognitive Processing Factors

Preinstructional strategies

Pretests

Advanced organizers

Are appropriate to:

Difficulty of the material

Ages of the learners

Prior knowledge (more unfamiliar, greater effect)

Length of the organizer

Whether organizes material to be learned

Clear, concise title at beginning of each unit in program

Instructional Text Format

Formatted for easy reading

Labels for headings identify different kinds of information

Sentence structure is easily understood for user population

Concept learning

Program uses a number of examples to teach each concept and foster generalizations

Program uses a number of nonexamples to teach each concept and to foster discrimination

Program presents rules and examples

Program presents a sequence of increasing difficulty

Practice in varied contexts is provided

Enhancing Retention

Vocabulary is appropriate for the user

When used, graphics are:

Embedded in the instructional content

Used for enhancement only (i.e. are not part of content but used to enhance presentation of feedback)

Graphics are used appropriately

Are relevant for user's age and level

Are not distracting

Support program's intents

Cues and/or prompt's are used after the wrong response

Action occurs on the screen

Program allows for:

Chunking

Depth of processing

Mnemonics

Difference/similarity distinctions

Enough practice to develop automatic response in appropriate situations;

Demonstration of exercise

Modeling

Adequate directions

Enhancing Attention

Critical features pointed out
Illustrations related to generality being taught

Enhancing Active Mental Processing

Use of rhetorical questions
Engage student in conversation
Require constructed responses.

Enhancing Metacognition

Learning strategy guidance overlayed on displays
Rule finding, transformation (means-end problem solving, and
rearrangement taught as needed

Affective Factors Which Influence Learning

Anxiety control factors

Keyboard anxiety is avoided by using a pointing device
Self-pacing is allowed

Motivational factors

INTEREST is aroused and sustained

- Appealing to perceptual level
- Appealing to information seeking needs
- Appealing to stable interests within people
- Appealing to interests in a particular situation

RELEVANCE-Learner perceives instruction as related to personal goals

- Opportunity for choice, responsibility, interpersonal influence
- Opportunity for no risk interaction

EXPECTANCY-Learner perceives likelihood of success

- Increased experience with success through instructional design strategies such as:
 - Advance organizers
 - Well stated objectives
- Personal control offered
- Attributional feedback provided
(ascribe to misconceptions not stupidity, Burton et al.)

SATISFACTION-Learner's intrinsic motivations and reactions to extrinsic rewards indicate satisfaction

- Task-endogenous rewards provided
- Unexpected rewards provided
- Verbal praise and informative feedback provided
- Feedback provided immediately following response
- Corrective feedback when immediately useful

GOALS

- Made known to the student
- Are proximal

AI Components

Knowledge representation

- The program uses semantic networks
- The program uses schemata
- The program uses context-dependent descriptions
- The program uses frame theory
- The program uses plans, episodes, stories within memory
- The program uses semantic framework of procedures
- The program uses abstract procedures and rationales
- The program uses ordering relationships
- The program uses production systems

Student mental modeling

- The program is alterable
- Alterable program content
- Alterable tutoring strategies
- Adoption of superior student methods

- The program has an expert model
- The program includes knowledge that could be misunderstood
- The program uses protocol analysis
- The program includes novice/expert differences in problem solving
 - Expert more directed
 - Expert more redirected
 - Expert plans
- The program uses a genetic graph
- Bug count and categorization is based on learning intent
- The program has a hierarchical organization of tasks
- The program has relationships and interrelationships among tasks
- The program maps from subtask requirements to implementation
- The program has a discrimination network

Tutoring capability

The program separates teaching and domain knowledge

The AI tutor simulates a good human tutor

Abilities of a good tutor

Generates instructional tasks

Answers free-form questions

Evaluates responses

Diagnoses bugs

Is able to learn

Skills of tutor

Knows subject matter

Alters teaching strategy

Uses natural language

Has world knowledge

Knows student so can model students

Knows higher level goal structures

Internal states and conceptual structures correspond to human

Gives advice appropriately -- neither too soon, too late

The tutor is capable of hypothesis evaluation and generation

The tutor uses 7 tutorial principles from GUIDON

1. Be clear

2. Provide orientation to new tasks by top-down orientation

3. Guide dialogue strictly

4. Account for incorrect behavior in terms of missing expertise

5. Probe student's misunderstanding

6. Provide assistance by methodologically introducing small steps

7. Examine student's understanding and introduce new

information when there is opportunity to do so

The tutor gives rule justifications

The tutor emphasizes strategy

The tutor understands pragmatics of language beyond literal meaning

The tutor makes explicit diagnostic strategy

The tutor makes explicit hierarchical data organization

Requirements for natural language processing

- The NLP demonstrates efficiency
(Response delays more than 2 sec. have a serious effect on performance of complex tasks via terminals.)
- The NLP demonstrates habitability
(Allow minor modifications to accepted sentence)
- The NLP is self-tutoring
- The NLP is aware of ambiguity

- The NLP uses Dyer's 11 metrics
- The NLP uses Dyer and Lennerts 14 retrieval functions

METHODOLOGY

Presentation

Methods for using the materials are:

- Specified by developer
- Left to user
- Left to teacher

Source

- Teacher's Guide
- Directions in program
- Other

Instructional Approach

Describe:

Teacher Use

Teacher training

- Specified
- Not mentioned

Teacher Use/Guidenace

- Inservice training
- Lesson plans in teacher's guide
- No guidance provided

Teacher's Manual

Provided

Not provided

Content provides:

- Enough information to familiarize techer with technical use program
- Classroom strategies for use
- Specific instructional activities to integrate program into curriculum

Instrucitons

- Are well organized
- Are clearly stated
- Provided step-by-step approach

Instructional strategies

Instructions in the progam for the sutdent are easily understood
Instructions in the program for the student re concise

User Control Over Rate and Sequence of Presentation

Time allowed for solving each problem

Rate of display presentation

Choice of sequence

Where entry must begin

Exiting activities

Reviewing instructions

Options available in sequence such as HELP, HUNT, ESCAPE

Means of Evaluation

Management System/Tests

CMI

- Used by program
- Not used

Functions implemented by CMI system

- Data collection and storage from learning situation
- Diagnosis
- Prescription
- Reporting

Tests

- Included in the program
- Are supplementary to print materials
- Not included

Test use is specified

- Pretest
- Unit test
- Diagnostic test
- Mastery test
- Other

Instructional strategies

Feedback

- Used after correct responses only
- Used after incorrect responses only
- Used after both types of responses
- Explains why the answer is wrong
- Informs what correct response is after -- number of attempts
- Responses are varied
 - From activity to activity
 - Within each activity

Uses

- Graphics
- Printed words
- Audio output
- Used appropriately
 - Nonthreatening
 - Immediate
 - Appropriately reinforces correct response
 - Appropriately reinforced wrong responses
- Remediates
- Is relevant for user's age and level

Records

- Are informative
- Are stored on magnetic device
- Are to be scored on printed sheets provided in the Guide

Instructional Design Congruence

Instructional Design Provisions

Intents match content
Intents match methodology
Intents match means of evaluation
Content matches methodology
Content matches means of evaluation
Methodology matches means of evaluation

Human Factors

Operator is oriented by:

- Location indicators
- Table of contents
- Menu
- Consistency
- Consideration of operator skill level
- Error check
- Input ease
- Editing

User can control

- Pacing
- Optional help
- Ability to escape
- Number of examples

Technical design allows

- Quick response time
- Quick loading time

Display Factors

Display theory

Each display is one of the following:

- Generality
- Example
- Generality practice
- Example practice
- Generality help
- Example help (attention focusing)
- Practice-feedback help

Display design

Each display is a unit

Black lettering is on white background

Thought units are formatted by lines

Right hand justification is avoided

Unneeded information is erased

A variety of styles is used

Each display is uncluttered

Instructional text format includes:

Short sentences

Space between lines of text

Clear screen before each new display

Avoid scrolling

Student control of pace

Graphics

Have clear resolution

Are in color

Are presented for maximum understanding

Are embedded in content (not used as reinforcer)

Include dynamic displays with one or more of the following:

- Timing

- Stress

- Animation

- Flashing

Cues, prompts used

Cursor or other highlighter directs attention to appropriate part of screen

Technical Considerations

Technical Considerations other than Displays

Audio

- Clear
- Realistic
- Contributes to program's value

Program Content

- Alterable by teacher
- Alterable by student
- Technically easy to utilize in computer

Random generation

- Used in practice
- Used in feedback

Tape/disks are accurate in

- Spelling
- Grammar
- Usage

Packaging

- Well designed for component parts
- Contains instructor's guide

Writing style

- Appropriate for user population

Content

- Accurate
- Free of bias

Consistent program design

Technical design

- Quick response time
- Quick loading

Development

Program has been field tested

Documentation of Learner-Verification and Revision is provided

No documentation and/or research has been provided

WEST Tutoring Principles
(Burton & Brown, 1982)

- Principle 1: Before giving advice, be sure the Issue used is one in which the student is weak.
- Principle 2: When illustrating an Issue, only use an Example (an alternative move) in which the result or outcome of that move is dramatically superior to the move made by the student.
- Principle 3: After giving the student advice, permit him to incorporate the Issue immediately by allowing him to repeat his turn.
- Principle 4: If a student is about to lose interrupt and tutor him only with moves that will keep him from losing.
- Principle 5: Do not tutor on two consecutive moves, no matter what.
- Principle 6: Do not tutor before the student has a chance to discover the game for himself.
- Principle 7: Do not provide only criticism when the Tutor breaks in! If the student makes an exceptional move, identify why it is good and congratulate him.
- Principle 8: After giving advice to the student, offer him a chance to retake his turn, but do not force him to.
- Principle 9: Always have the Computer Expert play an optimal game.
- Principle 10: If the student asks for help provide several levels of hints.
- Principle 11: If the student is losing consistently, adjust the level of play.
- Principle 12: If the student makes a potentially careless error, be forgiving. But provide explicit commentary in case it was not just careless.

WEST 1 MATERIALS

GAME PROCEDURES

1. Be sure student is seated comfortably and all game components are easy to see.
2. Give student a copy of the DIRECTIONS to read aloud. After each numbered item, ask if he/she understands. (Give reading assistance if necessary.) Point to game components as they are mentioned.
3. Answer all questions that pertain to the rules (e.g., 2nd player only gets 1 chance to tie the game; only respin the spinner in question; you can use the WORKSHEET as scratch paper; do not repeat numbers (or operations) unless the spinners give the same numbers; both people must agree on each answer or the player makes up a new number sentence; BUMPING is to the second town backwards from the opponent's position; you never go "up" shortcuts; you can't go past home.
4. If a question is a "what if ..." and a specific rule cannot be referenced, say:

"We'll take 3 practice turns to help you understand."

"You can figure that out as you play."

"The idea of the game is to figure that out as you play."
- 4a. Have student look at the KEYS - say:

"These are some helpful hints for playing the game - read each out loud".

After each one is read ask:

"Do you have any questions?" Refer as much as possible to the KEY items when answering - if too much detail is required say, "We'll be taking 3 practice turns to help you understand".
- 4b. Emphasize that the student may use the KEYS at any time during play, but you will not be giving explanations.
5. Three practice turns each: refer to each rule and go through the steps. If student violates a rule, or KEY, say "Look at rule ____ again." If student forgets to take a shortcut or jump a town, say "Look at rule ____ again; remember, I won't be able to help you when we start to play."
6. Use the worksheet and scoresheet during practice.

7. After Practice: remind student that he/she may refer to the directions, or KEYS, at any time but you will not be able to provide help. However, you will call the student on: moving to the wrong place, having a wrong number sentence, using wrong numbers or operations without any "penalty".
8. The MODEL for play is:
 - spin each spinner
 - record numbers
 - work out number sentence
 - show opponent
 - make move
9. For wrong answers, say "I don't agree with your answer - try again."
10. You may prompt the student to use the KEYS, saying, "Maybe the hints will help", but don't refer to specific items.
11. Record wins and ties on scoresheet. Play for a total of 45 minutes, including directions, practice and regular games. Conduct the interview and refer back to any questions raised during the game requiring answers, if necessary.

For each student:

Pre-game:	directions KEYS questions practice (model playing sequence)
In-game:	model game behavior use worksheet use scoresheet
Post-game:	interview observation gather all sheets together and clip (worksheet, scoresheet, interview & ratings)

WEST DIRECTIONS

Object of the Game: Starting at 0, be the first player to reach HOME (70).

Rules of the Game:

- ___ 1. Newest player may choose to go 1st or 2nd. Players then take turns. The 2nd player gets the last turn to possibly tie the game.
- ___ 2. Player spins each spinner once and writes the three numbers on the PLAYER WORKSHEET. Use the number each spinner is closest to or respin if necessary.
- ___ 3. Use the three numbers in a number sentence to determine your move. You may not use a spinner number or an operation more than once. (Spinners pointing to the same numbers are okay to use.)
- ___ 4. Work out your answer on the PLAYER WORKSHEET and show the complete number sentence to your opponent.
- ___ 5. If you and your opponent agree on the answer, move that number of spaces on the WEST board. If you don't agree you have one more chance to get a right answer.
- ___ 6. If you land on a TOWN you may jump to the next TOWN.
- ___ 7. If you land on the top of a SHORTCUT line, you may move to the space at the bottom of the line.
- ___ 8. If you land on your opponent, BUMP him/her back two towns.
- ___ 9. In order to win, you must have the exact number to land on HOME.

Date _____

Time _____

PLAYER WORKSHEET

Spinner Numbers

1. _____
2. _____
3. _____
4. _____
5. _____
6. _____
7. _____
8. _____
9. _____
10. _____
11. _____
12. _____
13. _____
14. _____
15. _____
16. _____
17. _____
18. _____
19. _____
20. _____
21. _____
22. _____
23. _____
24. _____

Number Sentences

Name _____

Date _____

QUESTIONNAIRE

Below are some questions about playing games. Circle the number that most closely describes how you feel. (There are no right or wrong answers.)

1. How much do you like to play board games?

not much		some		a lot
1	2	3	4	5

2. How often do you play board games?

hardly ever		sometimes		very often
1	2	3	4	5

3. Are you good at playing board games?

not good		fair		very good
1	2	3	4	5

4. How much do you think people learn by playing games?

nothing		some		a lot
1	2	3	4	5

5. How much do you like to play games that involve math?

not much		some		a lot
1	2	3	4	5

6. How good are you at math?

not good			fair		very good
1	2	3	4	5	

7. How much do you like to receive help when you do math work?

not much			some		a lot
1	2	3	4	5	

8. How much do you like to receive help during a game?

not much			some		a lot
1	2	3	4	5	

9. How much do you want to know about a game before you play it?

not much			some		a lot
1	2	3	4	5	

10. How important do you think it is to read a game's directions?

not important			somewhat important		very important
1	2	3	4	5	

STOP! Do not turn the page!

Name _____

MATHEMATICS PRETEST

PART I. ADDITION AND SUBTRACTION FACTS

DIRECTIONS: You will be given 2 minutes to solve the number facts below. When told to begin, write the answers on the lines provided.

ADDITION

- | | | |
|---------------------|---------------------|---------------------|
| 1) $3 + 2 =$ _____ | 11) $6 + 6 =$ _____ | 21) $4 + 4 =$ _____ |
| 2) $4 + 8 =$ _____ | 12) $8 + 5 =$ _____ | 22) $7 + 9 =$ _____ |
| 3) $0 + 5 =$ _____ | 13) $2 + 9 =$ _____ | 23) $8 + 2 =$ _____ |
| 4) $6 + 9 =$ _____ | 14) $3 + 6 =$ _____ | 24) $5 + 3 =$ _____ |
| 5) $7 + 4 =$ _____ | 15) $9 + 4 =$ _____ | 25) $9 + 1 =$ _____ |
| 6) $2 + 5 =$ _____ | 16) $7 + 8 =$ _____ | |
| 7) $1 + 8 =$ _____ | 17) $3 + 3 =$ _____ | |
| 8) $3 + 9 =$ _____ | 18) $7 + 3 =$ _____ | |
| 9) $7 + 0 =$ _____ | 19) $8 + 6 =$ _____ | |
| 10) $5 + 9 =$ _____ | 20) $2 + 7 =$ _____ | |

SUBTRACTION

- | | |
|----------------------|----------------------|
| 1) $5 - 2 =$ _____ | 11) $9 - 3 =$ _____ |
| 2) $11 - 3 =$ _____ | 12) $2 - 0 =$ _____ |
| 3) $6 - 3 =$ _____ | 13) $5 - 0 =$ _____ |
| 4) $10 - 3 =$ _____ | 14) $10 - 2 =$ _____ |
| 5) $1 - 2 =$ _____ | 15) $8 - 5 =$ _____ |
| 6) $9 - 1 =$ _____ | 16) $7 - 2 =$ _____ |
| 7) $12 - 3 =$ _____ | 17) $0 - 5 =$ _____ |
| 8) $7 - 7 =$ _____ | 18) $9 - 7 =$ _____ |
| 9) $12 - 6 =$ _____ | 19) $3 - 6 =$ _____ |
| 10) $11 - 2 =$ _____ | 20) $12 - 4 =$ _____ |

STOP! Do not turn the page!

PART II MULTIPLICATION & DIVISION FACTS

DIRECTIONS: You will be given 2 minutes to solve the number facts below. When told to begin, write the answers on the lines provided.

MULTIPLICATION

- | | | |
|--------------------------------------|--------------------------------------|--------------------------------------|
| 1) $1 \times 3 = \underline{\quad}$ | 11) $6 \times 7 = \underline{\quad}$ | 21) $4 \times 3 = \underline{\quad}$ |
| 2) $5 \times 6 = \underline{\quad}$ | 12) $9 \times 2 = \underline{\quad}$ | 22) $6 \times 1 = \underline{\quad}$ |
| 3) $2 \times 2 = \underline{\quad}$ | 13) $4 \times 7 = \underline{\quad}$ | 23) $4 \times 4 = \underline{\quad}$ |
| 4) $5 \times 1 = \underline{\quad}$ | 14) $1 \times 8 = \underline{\quad}$ | 24) $5 \times 2 = \underline{\quad}$ |
| 5) $8 \times 3 = \underline{\quad}$ | 15) $7 \times 7 = \underline{\quad}$ | 25) $0 \times 7 = \underline{\quad}$ |
| 6) $2 \times 6 = \underline{\quad}$ | 16) $2 \times 3 = \underline{\quad}$ | 26) $4 \times 6 = \underline{\quad}$ |
| 7) $1 \times 1 = \underline{\quad}$ | 17) $4 \times 0 = \underline{\quad}$ | 27) $5 \times 5 = \underline{\quad}$ |
| 8) $5 \times 7 = \underline{\quad}$ | 18) $9 \times 3 = \underline{\quad}$ | 28) $7 \times 2 = \underline{\quad}$ |
| 9) $8 \times 2 = \underline{\quad}$ | 19) $4 \times 5 = \underline{\quad}$ | 29) $0 \times 0 = \underline{\quad}$ |
| 10) $3 \times 3 = \underline{\quad}$ | 20) $6 \times 6 = \underline{\quad}$ | 30) $6 \times 5 = \underline{\quad}$ |

DIVISION

- | | | | |
|------------------------------------|-------------------------------------|-------------------------------------|-------------------------------------|
| 1) $3 \div 1 = \underline{\quad}$ | 6) $12 \div 6 = \underline{\quad}$ | 11) $8 \div 1 = \underline{\quad}$ | 16) $10 \div 2 = \underline{\quad}$ |
| 2) $8 \div 2 = \underline{\quad}$ | 7) $1 \div 1 = \underline{\quad}$ | 12) $6 \div 3 = \underline{\quad}$ | 17) $0 \div 7 = \underline{\quad}$ |
| 3) $4 \div 2 = \underline{\quad}$ | 8) $16 \div 2 = \underline{\quad}$ | 13) $0 \div 1 = \underline{\quad}$ | 18) $14 \div 2 = \underline{\quad}$ |
| 4) $25 \div 5 = \underline{\quad}$ | 9) $9 \div 3 = \underline{\quad}$ | 14) $12 \div 4 = \underline{\quad}$ | 19) $49 \div 7 = \underline{\quad}$ |
| 5) $24 \div 3 = \underline{\quad}$ | 10) $18 \div 2 = \underline{\quad}$ | 15) $6 \div 6 = \underline{\quad}$ | 20) $5 \div 5 = \underline{\quad}$ |

STOP! Do not turn the page!

PART III. FAMILIES OF FACTS - ADDITION & SUBTRACTION

DIRECTIONS: Each set of four number sentences below contains related addition and subtraction facts. Fill in the missing numbers in each family of facts.

SAMPLE

$$2 + \underline{\quad} = 5$$

$$3 + 2 = 5$$

$$5 - 2 = \underline{\quad}$$

$$\underline{\quad} - 3 = 2$$

$$1) \quad 6 + 4 = \underline{\quad}$$

$$\underline{\quad} + 6 = 10$$

$$10 - 4 = 6$$

$$10 - \underline{\quad} = 4$$

$$4) \quad \underline{\quad} + 4 = \underline{\quad}$$

$$\underline{\quad} + 5 = 9$$

$$9 - \underline{\quad} = 4$$

$$9 - 4 = \underline{\quad}$$

$$2) \quad 6 + 7 = 13$$

$$7 + 6 = \underline{\quad}$$

$$\underline{\quad} - 7 = 6$$

$$13 - 6 = \underline{\quad}$$

$$5) \quad 6 + \underline{\quad} = \underline{\quad}$$

$$\underline{\quad} + \underline{\quad} = \underline{\quad}$$

$$\underline{\quad} - 6 = 5$$

$$\underline{\quad} - 5 = 6$$

$$3) \quad \underline{\quad} + 1 = 6$$

$$1 + 5 = \underline{\quad}$$

$$\underline{\quad} - \underline{\quad} = 1$$

$$\underline{\quad} - 1 = 5$$

$$6) \quad \underline{\quad} + \underline{\quad} = \underline{\quad}$$

$$\underline{\quad} + 7 = \underline{\quad}$$

$$12 - \underline{\quad} = 7$$

$$12 - 7 = \underline{\quad}$$

PART IV. FAMILIES OF FACTS - MULTIPLICATION & DIVISION

DIRECTIONS: Each set of four number sentences below contains related multiplication and division facts. Fill in the missing numbers in each family of facts.

SAMPLE

$4 \times 2 = \underline{\quad}$
$2 \times \underline{\quad} = 8$
$\underline{\quad} \div 2 = 4$
$8 \div 4 = \underline{\quad}$

1) $3 \times 6 = 18$ $6 \times 3 = \underline{\quad}$ $\underline{\quad} \div 6 = 3$ $18 \div 3 = \underline{\quad}$	4) $\underline{\quad} \times 3 = \underline{\quad}$ $\underline{\quad} \times 9 = 27$ $27 \div \underline{\quad} = 3$ $27 \div 3 = \underline{\quad}$
2) $7 \times \underline{\quad} = 28$ $4 \times 7 = 28$ $28 \div 4 = \underline{\quad}$ $\underline{\quad} \div 7 = 4$	5) $\underline{\quad} \times 4 = 24$ $4 \times \underline{\quad} = \underline{\quad}$ $24 \div 4 = \underline{\quad}$ $\underline{\quad} \div \underline{\quad} = \underline{\quad}$
3) $7 \times \underline{\quad} = 35$ $5 \times 7 = \underline{\quad}$ $\underline{\quad} \div \underline{\quad} = 5$ $\underline{\quad} \div 5 = 7$	6) $3 \times \underline{\quad} = \underline{\quad}$ $\underline{\quad} \times \underline{\quad} = \underline{\quad}$ $\underline{\quad} \div 3 = 5$ $\underline{\quad} \div 5 = 3$

PART V. FACTORS OF NUMBERS

DIRECTIONS: A number can be divided evenly by each of its FACTORS. For example, the FACTORS of 8 are 1, 2, 4, 8 because these numbers divide 8 evenly. Fill in the missing factors for the numbers given below.

Example: 8: 1, 2, 4, 8

- 1) 12: 1, __, __, 4, 6, 12
- 2) 16: 1, 2, __, __, 16
- 3) 20: 1, __, 4, __, __, 20
- 4) 30: 1, 2, __, 5, __, __, 15, 30

PART VI. NUMBER SENTENCES

A. DIRECTIONS: Solve each number sentence below. Write the answers on the lines provided.

- | | |
|--------------------------------|--------------------------------|
| 1) $3 \times 3 + 6 =$ ____ | 11) $6 \div 2 + 3 =$ ____ |
| 2) $1 \times 2 - 5 =$ ____ | 12) $5 \div 1 - 3 =$ ____ |
| 3) $1 \times 1 \div 1 =$ ____ | 13) $4 + 6 - 2 =$ ____ |
| 4) $6 \div 3 \times 2 =$ ____ | 14) $4 - 1 + 6 =$ ____ |
| 5) $2 \times 4 + 5 =$ ____ | 15) $4 \times 3 - 4 =$ ____ |
| 6) $3 \times 3 \div 3 =$ ____ | 16) $4 + 7 - 2 =$ ____ |
| 7) $6 - 2 + 0 =$ ____ | 17) $3 - 3 + 2 =$ ____ |
| 8) $7 \times 1 + 4 =$ ____ | 18) $7 - 0 + 3 =$ ____ |
| 9) $3 \times 4 - 6 =$ ____ | 19) $6 - 1 + 3 =$ ____ |
| 10) $4 \div 4 \times 2 =$ ____ | 20) $4 \times 6 \div 3 =$ ____ |

B. DIRECTIONS: Circle the correct answer for each number sentence below.

- | | | | |
|----------------------------|-------|-------|------------|
| 1) $3 + 4 \times 2 =$ | a) 14 | b) 11 | c) neither |
| 2) $2 - 1 \times 2 =$ | a) 0 | b) 2 | c) neither |
| 3) $4 \div (2 \times 2) =$ | a) 4 | b) 1 | c) neither |
| 4) $3 - 0 \div 3 =$ | a) 1 | b) 3 | c) neither |
| 5) $2 - (2 + 4) =$ | a) 4 | b) 0 | c) neither |
| 6) $6 \div 3 \times 2 =$ | a) 1 | b) 4 | c) neither |
| 7) $3 - 2 + 1 =$ | a) 0 | b) 1 | c) neither |
| 8) $(3 + 3) \div 3 =$ | a) 2 | b) 4 | c) neither |

Name _____

OBSERVATIONAL RATINGS

Comments:

1. The student was enthusiastic:

rarely or
not at all

most of
the time

1

2

3

4

2. The student displayed effort:

rarely or
not at all

most of
the time

1

2

3

4

3. The student reacted to losing by:

being
discouraged

being
challenged

1

2

3

4

4. The student referred to the "KEYS":

rarely or
not at all

very
often

1

2

3

4

5. The student's overt behavior included:

few questions

many questions

1

2

3

4

Comments:

6. The student's behavior included:

few comments

many comments

1

2

3

4

7. The student's general playing strategy indicated:

reflective

impulsive

1

2

3

4

8. As the student played, his/her playing ability:

stayed about
the same

improved
a lot

1

2

3

4

9. In general, the math ability displayed was:

poor

excellent

1

2

3

4

10. In general, the game playing ability was:

poor

1

2

3

4

MATHEMATICS POSTTEST

PART I. SOLVING NUMBER SENTENCES

DIRECTIONS: Solve each number sentence below. Put the answers on the lines provided.

- | | |
|----------------------------------|-----------------------------------|
| 1) $(1 - 1) \times 5 =$ _____ | 9) $6 \div (0 + 2) =$ _____ |
| 2) $7 \times 0 + 3 =$ _____ | 10) $4 - 2 \times 2 =$ _____ |
| 3) $(2 + 6) \div 4 =$ _____ | 11) $6 \div 3 \times 2 =$ _____ |
| 4) $2 \times 3 - 6 =$ _____ | 12) $3 + 0 - 5 =$ _____ |
| 5) $6 - (4 + 3) =$ _____ | 13) $7 \div (4 + 3) =$ _____ |
| 6) $4 + 6 \div 2 =$ _____ | 14) $7 - 4 + 2 =$ _____ |
| 7) $0 \times (3 \div 3) =$ _____ | 15) $2 \times 6 \div 4 =$ _____ |
| 8) $2 \div 2 - 4 =$ _____ | 16) $4 \div (1 \times 2) =$ _____ |

PART II. WRITING NUMBER SENTENCES

DIRECTIONS: In each section below you will be writing number sentences. Be sure that you do not use an operation twice in one number sentence - for example, using 2, 4 and 7,

$$2 \times 7 + 4 = 18 \quad \text{is OK}$$

$$2 \times 7 \times 4 = 56 \quad \text{is NOT OK}$$

NOTE: You may change the order of the numbers and you may use parentheses if desired.

A. Write a number sentence with each set of numbers below that equals the LARGEST possible answer:

1) 1, 2, and 5

2) 3, 0, and 4

3) 2, 4, and 3

4) 3, 7, and 1

B. Write a number sentence for each problem below or write "IMPOSSIBLE" if it cannot be done:

1) Use 2, 2 and 7 to equal 12

2) Use 1, 4 and 6 to equal -2

3) Use 2, 4 and 5 to equal 10

4) Use 5, 2 and 4 to equal 8

5) Use 6, 3 and 1 to equal 15

6) Use 1, 3 and 5 to equal 6

7) Use 2, 6 and 4 to equal 0

C. Arrange the numbers below into number sentences that are all EQUAL:

1) 3, 0, and 2

2) 1, 1, and 7

3) 2, 4, and 6

PART III. MAKING DIFFERENT NUMBER SENTENCES

DIRECTIONS: Use 2, 1 and 6 in number sentences to get as many different answers as you can. Do not use an operation twice in the same sentence. You may change the order of the numbers and use parentheses if desired.

Use only the numbers 2, 1 and 6.

STRATEGY POSTTEST

DIRECTIONS: For each game board shown on the following pages -


1. Choose the move you would make for the stagecoach.
2. Write the number sentence you would use by the words "Your Move".
3. Circle the place where the stagecoach would land.

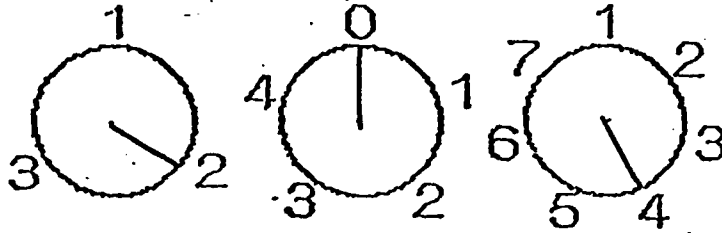
The first page is done for you.

REMEMBER! Do not use an operation twice in the same number sentence.

SAMPLE

① Look for stagecoach and select a move

Stagecoach's turn 

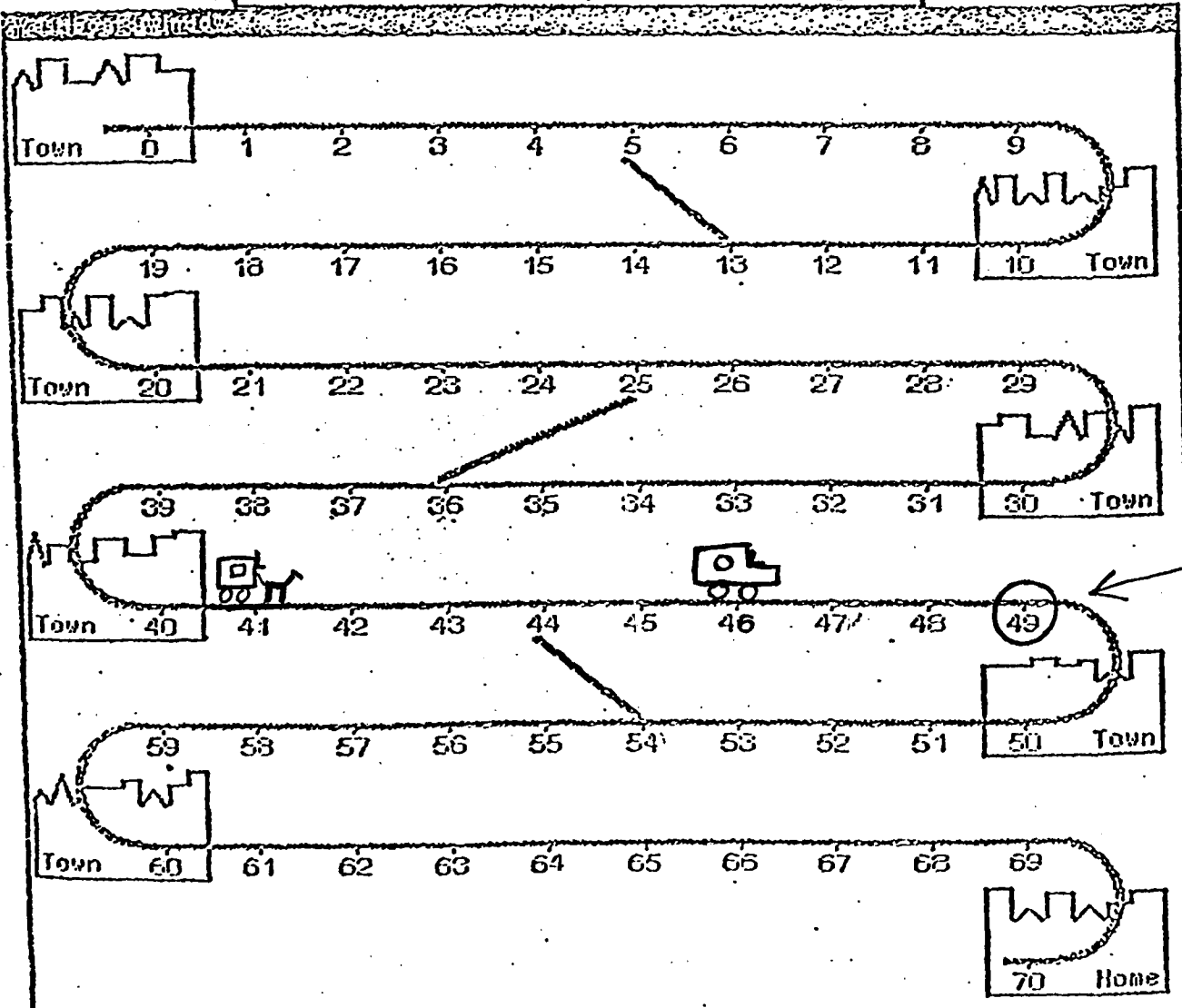


The numbers are:

2 0 4

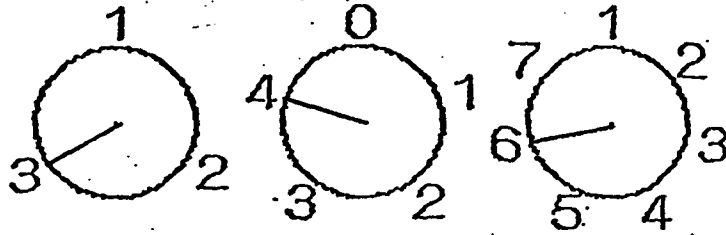
YOUR MOVE: $4 \times 2 + 0 = 8$

② Write your number sentence



③ Circle where you will land

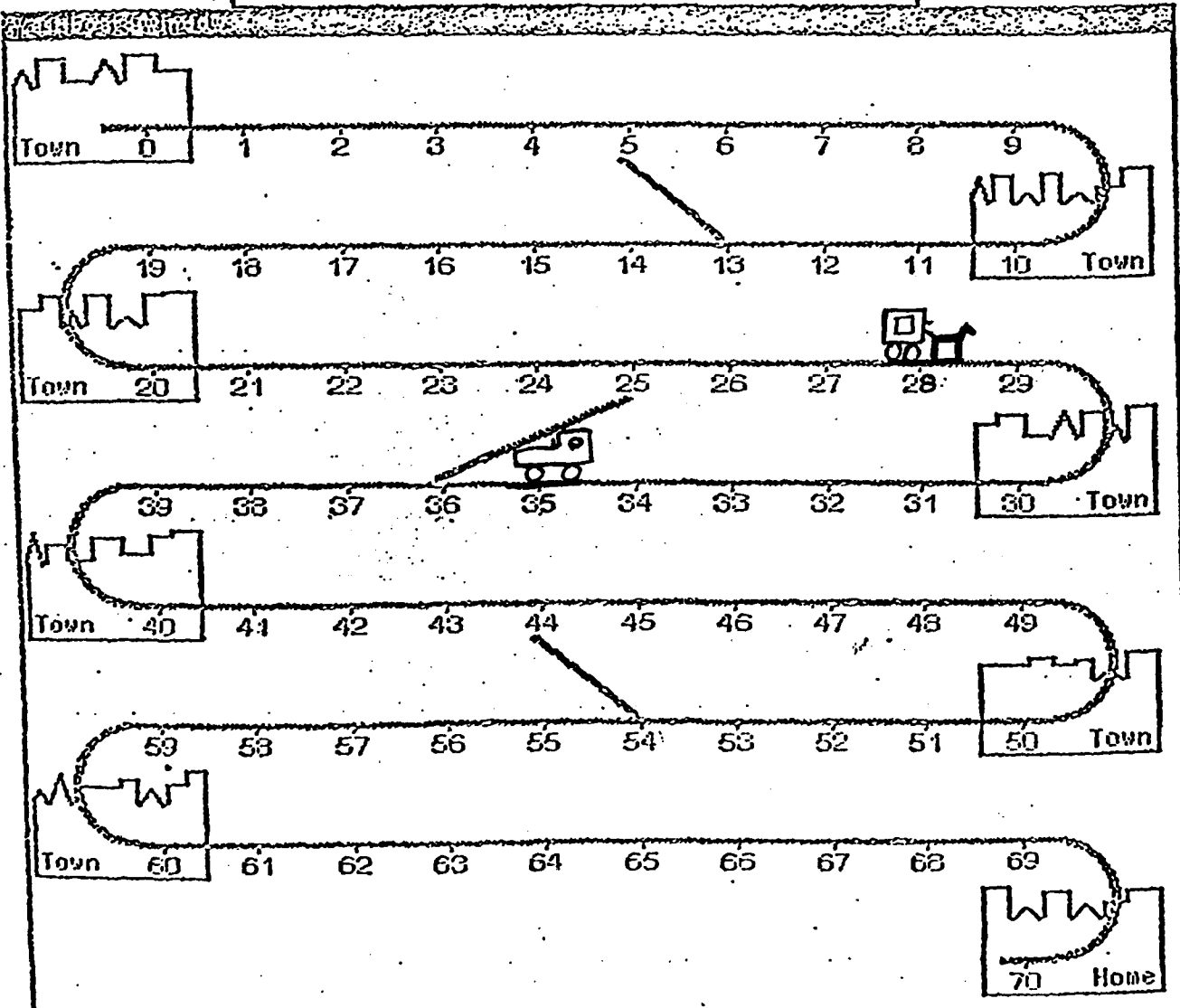
Stagecoach's turn



The numbers are:

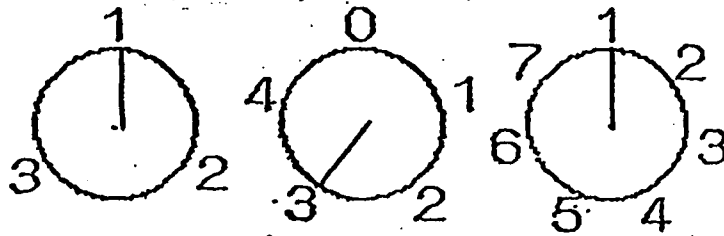
3 4 6

YOUR MOVE:



PAGE 3

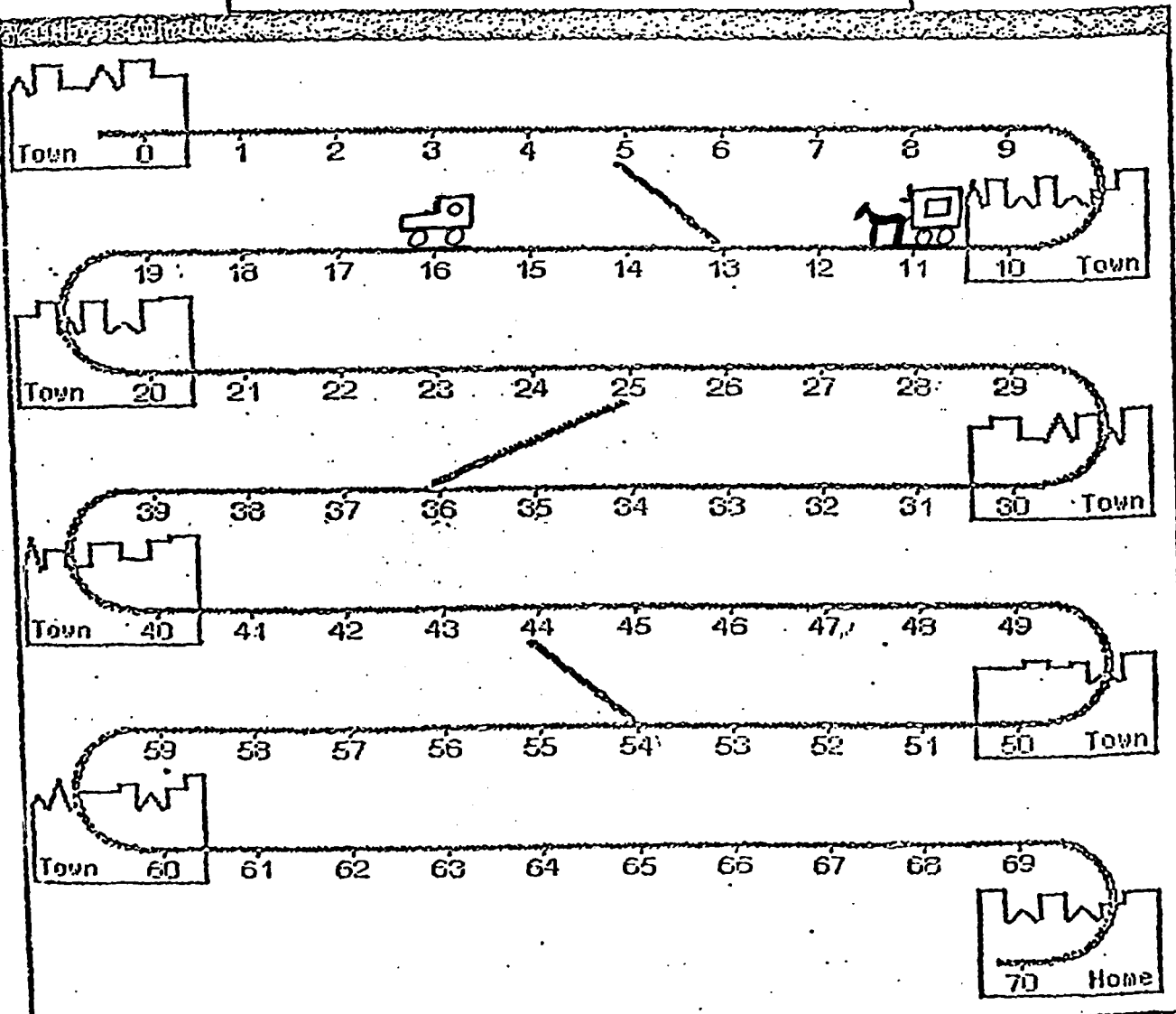
Stagecoach's turn



The numbers are:

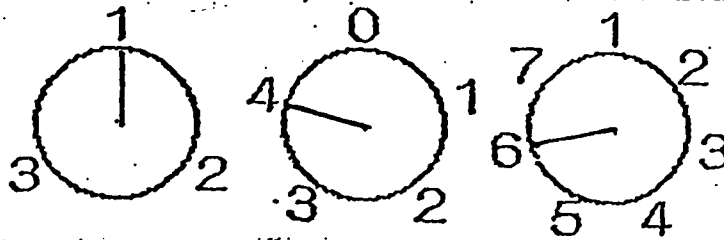
1 3 1

YOUR MOVE:



PAGE 4

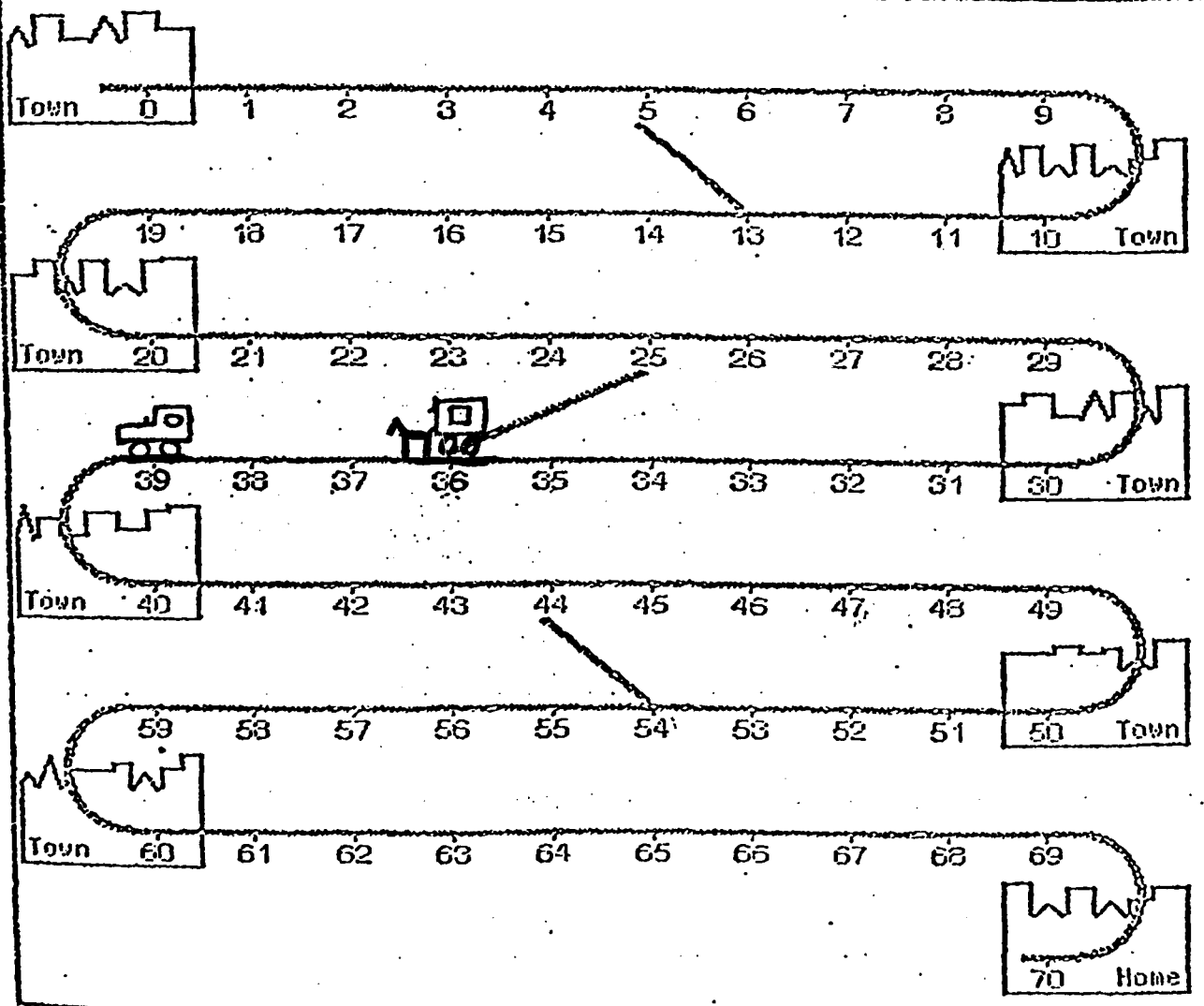
Stagecoach's turn



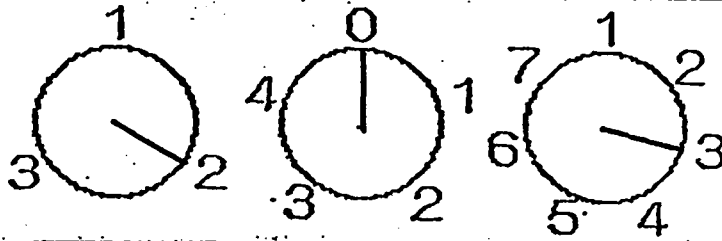
The numbers are:

1 4 6

YOUR MOVE:



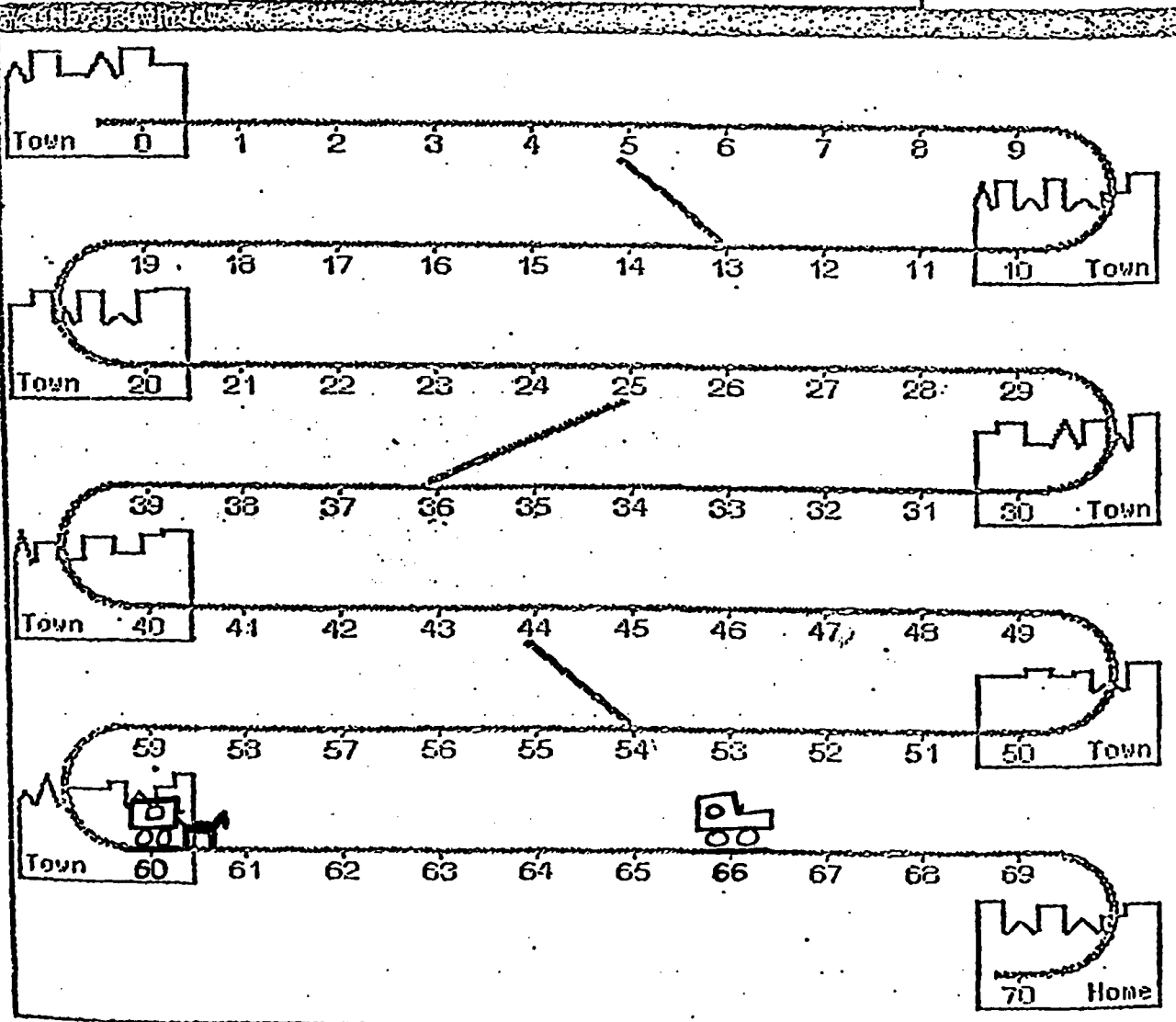
Stagecoach's turn



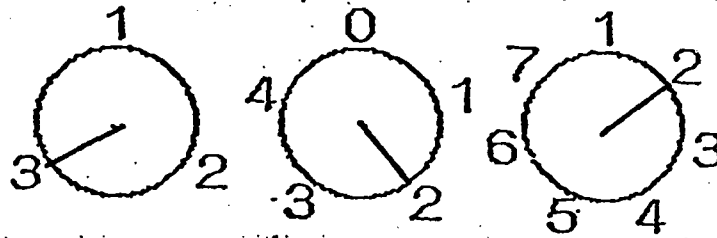
The numbers are:

2 0 3

YOUR MOVE:



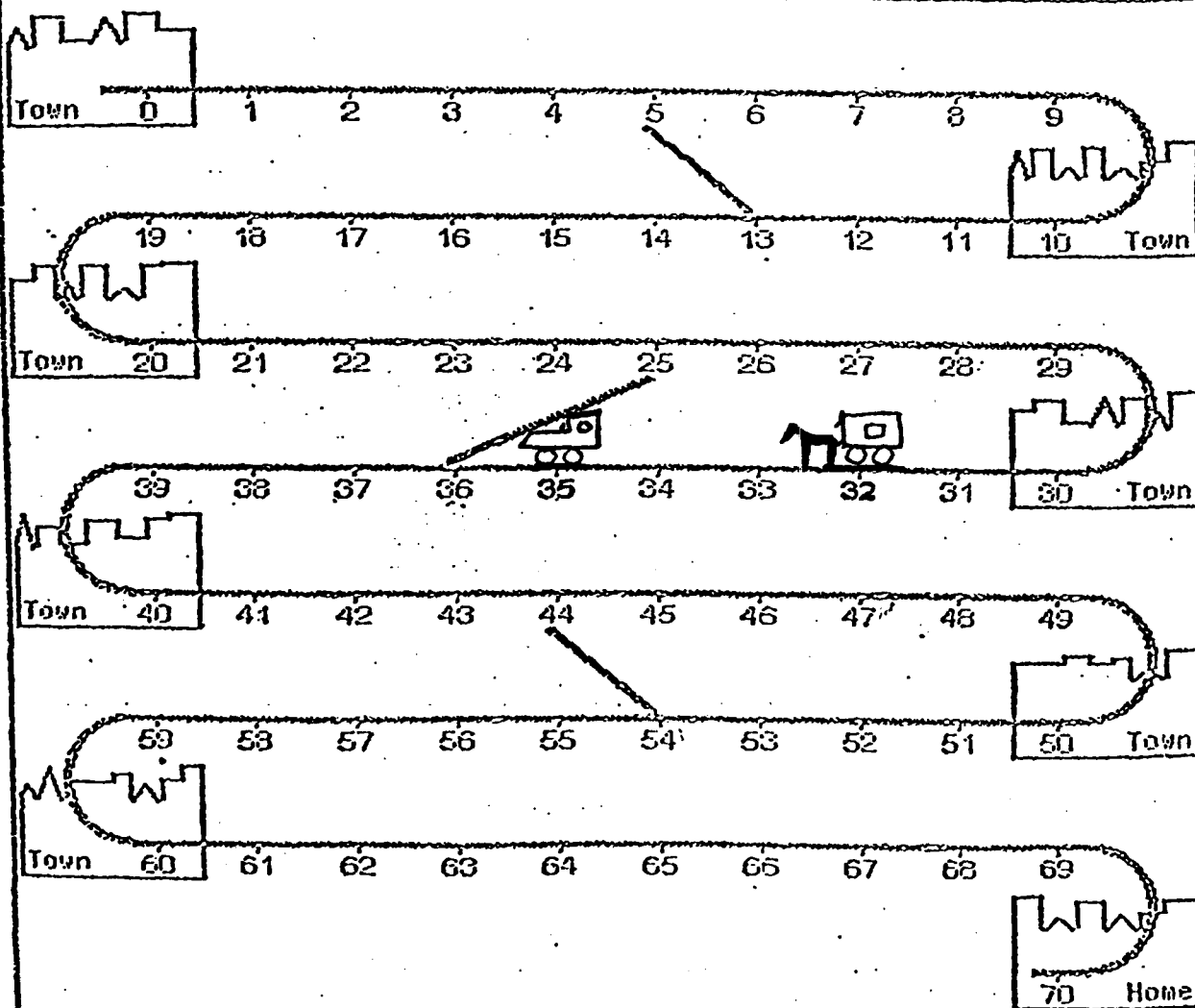
Stagecoach's turn



The numbers are:

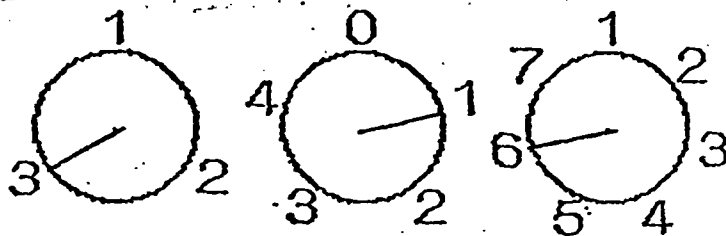
3 2 2

YOUR MOVE:



PAGE 7

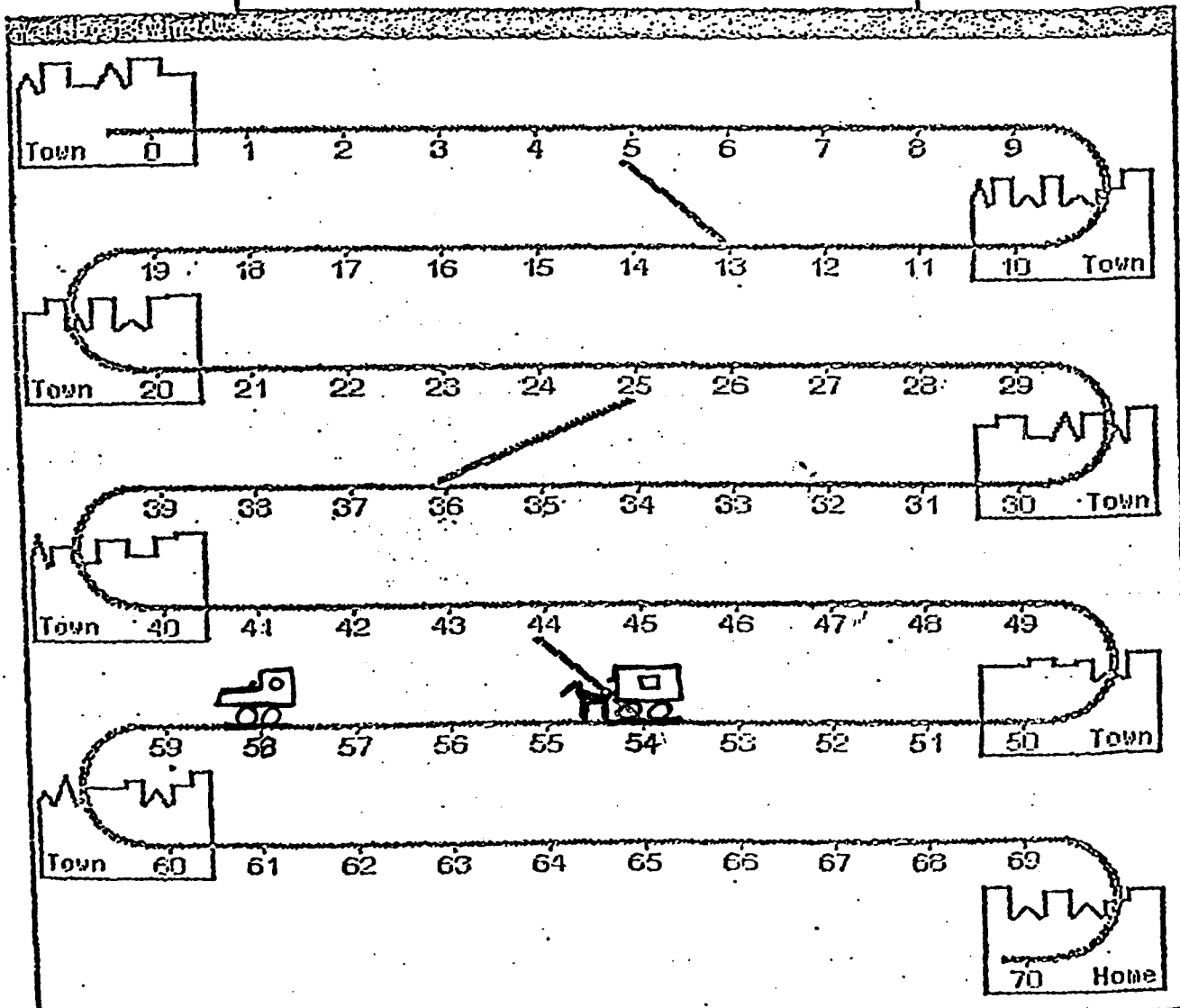
Stagecoach's turn



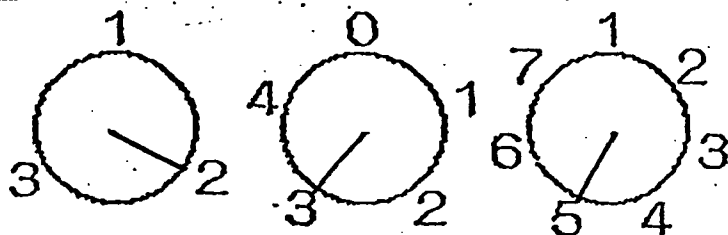
The numbers are:

3 1 6

YOUR MOVE:



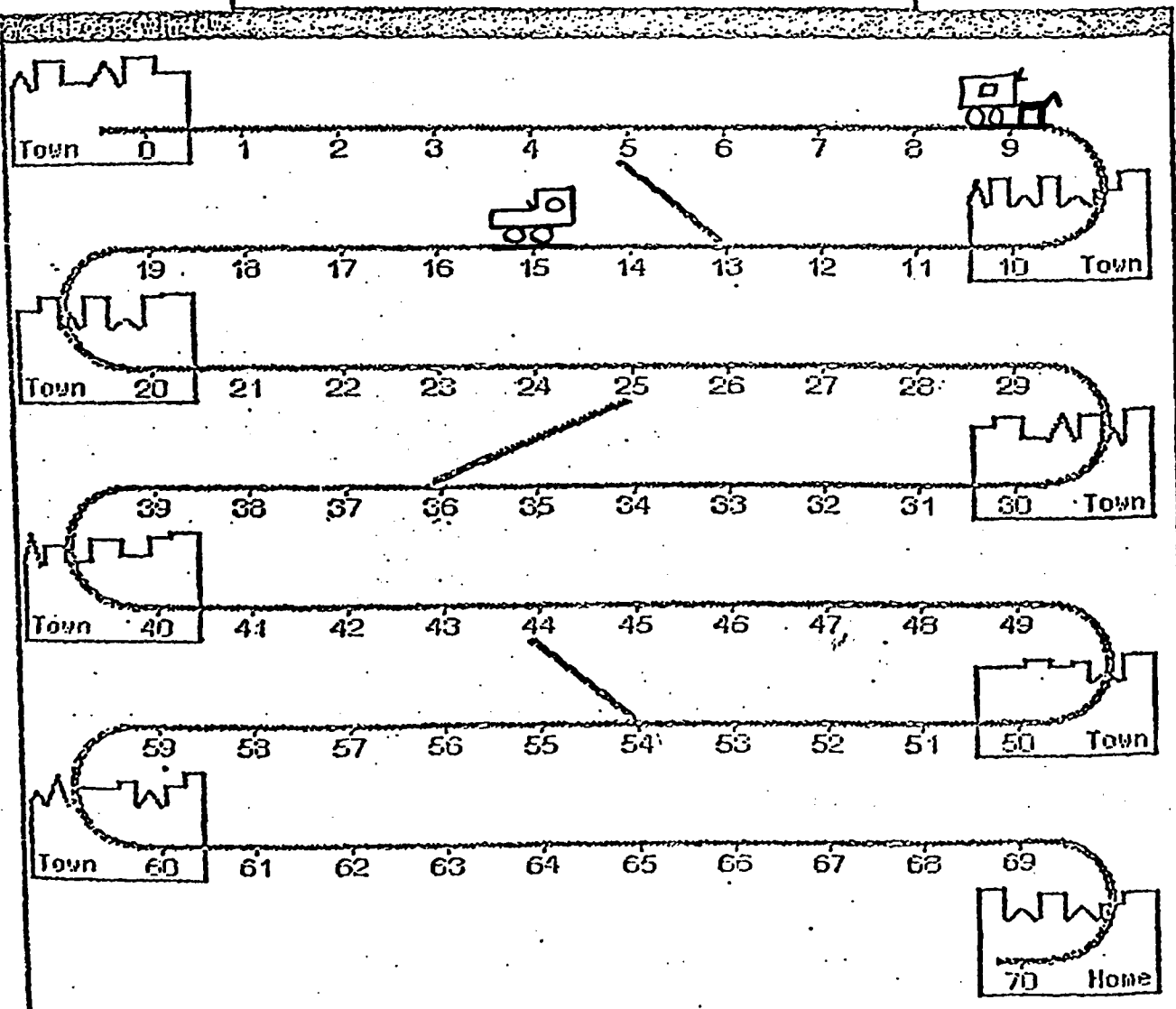
Stagecoach's turn



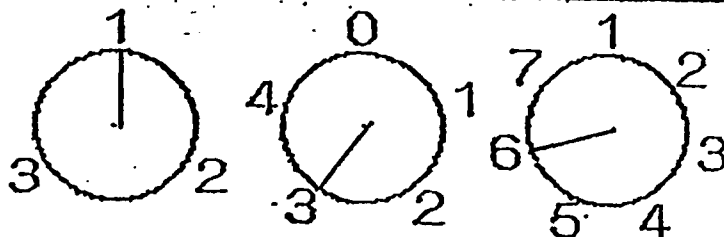
The numbers are:

2 3 5

YOUR MOVE:



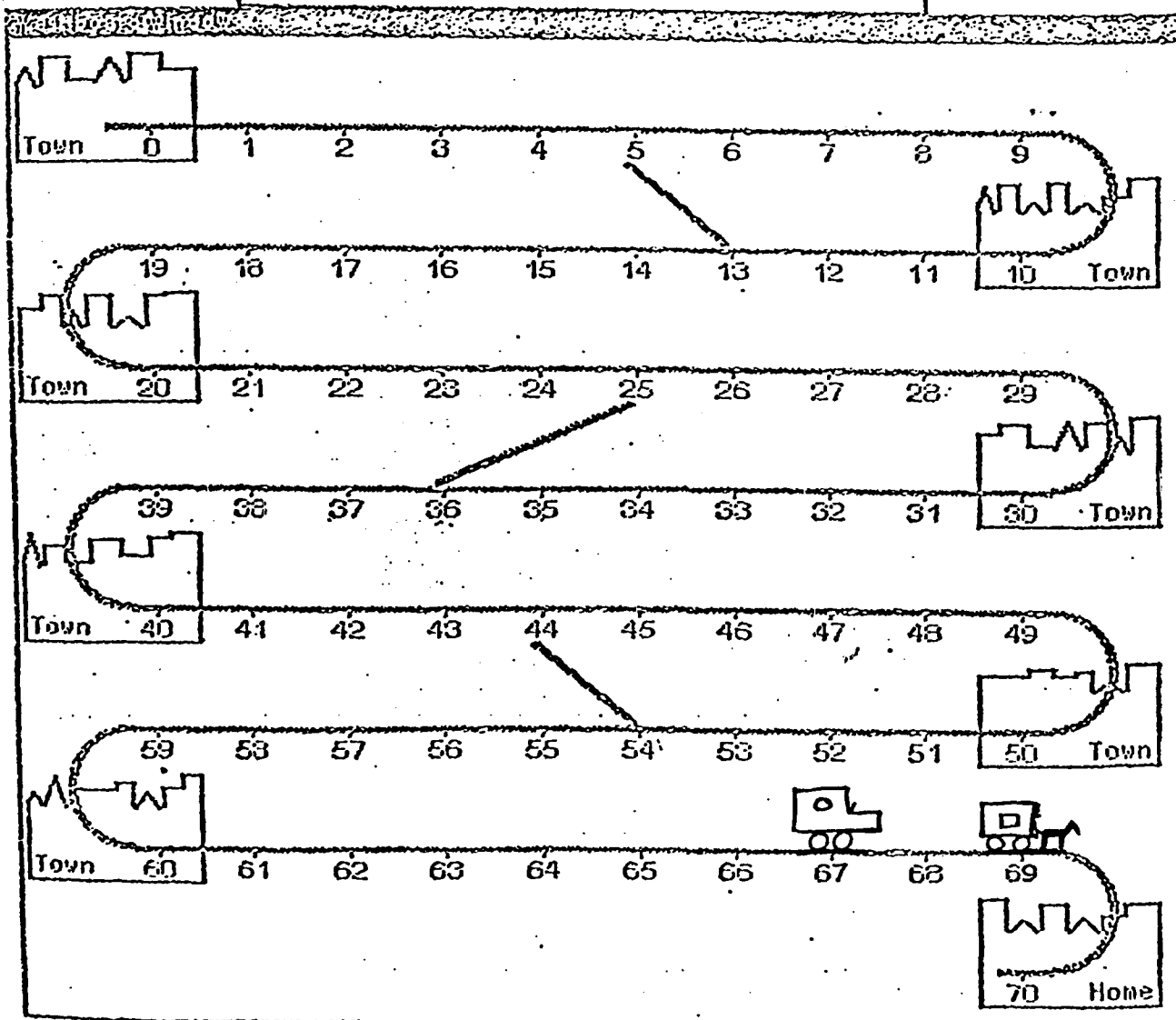
Stagecoach's turn



The numbers are:

1 3 6

YOUR MOVE:



Name _____

INTERVIEW

1. Did you enjoy playing the game? Why or why not?
2. What was the hardest part of the game for you?
3. What was the easiest part of the game for you?
4. Did you learn anything from playing the game? What?
5. What would you change about the game? Why?

6. Did the game help you become better at math? In what way?

7. How would you explain the important rules in the game to a new player?

[Skip to 13 for no checklist group.]

8. Did you need the "Keyus". In what way?

9. Did you use the "Keys" often? When?

10. What was most helpful about the "Keys"?

11. What was least helpful about the "Keys"?

12. Could you have played as well without the "Keys"? Why or why not?

13. Would you have liked to have had help on anything else? What in particular?

14. Any other comments?

WEST 2 MATERIALS

EXPERIMENTER'S TRAINING NOTES

(FOR EXPERIMENTAL Ss ONLY...CONTROL GP DOES NOT GET TRAINING)

"We are studying how students can learn to play a math game called WEST on a special kind of computer---an intelligent one. Tomorrow (this afternoon---for the first 2 Ss) you will have the chance to play WEST with the computer."

"First, I'd like to ask you some questions about games, computers & math."

GIVE PRE-NAME INTERVIEW. PUT INTERVIEW PAGE IN S's FOLDER.

"Now we will give you a very short math paper to do. You will have 5 minutes to finish as much as you can. Do your best, and raise your hand if you have a question."

GIVE PRETEST. ALLOW 5 MIN TOTAL. TELL Ss WHEN 1 MINUTE REMAINS. PUT FINISHED PRETESTS IN Ss' FOLDERS.

"Now you will have about 30 minutes to read this booklet. It has some important information in it that will help you play the game better. I will let you know when half of the time is past."

GIVE TRAINING BOOKLET. FOLLOW SCHEDULE SO THAT Ss GET CORRECT TRAINING BOOKLET. (MATH OR MATH/STRATEGY). ALLOW 30 MIN. TOTAL. TELL Ss WHEN 15 MIN IS PAST. WHEN Ss ASK QUESTIONS, REFER THEM TO APPROPRIATE DIRECTIONS INBOOKLET. AVOID GIVING SUBSTANTIVE ANSWERS. PUT FINISHED BOOKLETS IN Ss' FOLDERS.

"That's all for now. Tomorrow (or "this afternoon" for first 2 Ss) you will get to play WEST on the computer. Please don't talk to your friends about the booklet or the game until next week."

ESCORT Ss BACK TO CLASS.

NAME _____

PRE-GAME INTERVIEW

READ THIS TO STUDENTS AND CIRCLE THE NUMBER THAT CORRESPONDS TO THEIR ANSWER.

"I would like to ask you some questions about playing games, and you can tell me how you feel about them. There aren't any "right" or "wrong" answers. Use this 5-point scale to tell me how you feel about these things. 1 = NOT AT ALL, 3 = SOME, 5 = A LOT."

	NOT AT ALL		SOME		A LOT
1 How good are you at playing games?	1	2	3	4	5
2 How much do you like to play games that involve math?	1	2	3	4	5
3 How often do you play games on a computer?	1	2	3	4	5
4 How good are you at math?	1	2	3	4	5
5 How much do you like to receive help when you do math?	1	2	3	4	5
6 How much do you like to receive help when you play a game?	1	2	3	4	5
7 How much do you like to know about a game before you play it?	1	2	3	4	5
8 How important do you think it is to read the directions for a game?	1	2	3	4	5
9 How much do you like to use computers?	1	2	3	4	5

Name _____

MATHEMATICS PRETEST

DIRECTIONS: You will be given 5 minutes to solve the number problems below. When told to begin, write the answers on the lines provided.

ADDITION

1) $3 + 2 =$ _____

3) $8 + 5 =$ _____

5) $7 + 9 =$ _____

2) $7 + 0 =$ _____

4) $4 + 4 =$ _____

SUBTRACTION

1) $10 - 3 =$ _____

4) $8 - 5 =$ _____

2) $12 - 6 =$ _____

5) $12 - 4 =$ _____

3) $5 - 0 =$ _____

MULTIPLICATION

1) $1 \times 8 =$ _____

4) $6 \times 6 =$ _____

2) $4 \times 0 =$ _____

5) $7 \times 2 =$ _____

3) $9 \times 3 =$ _____

DIVISION

1) $8 \div 2 =$ _____

4) $49 \div 7 =$ _____

2) $12 \div 6 =$ _____

5) $5 \div 5 =$ _____

3) $16 \div 2 =$ _____

PART III. FAMILIES OF FACTS - ADDITION & SUBTRACTION

Fill in the missing numbers in each family of facts.

1) $6 + 4 = \underline{\quad}$

2) $\underline{\quad} + 6 = 10$

3) $10 - 4 = 6$

4) $10 - \underline{\quad} = 4$

PART IV. FAMILIES OF FACTS - MULTIPLICATION & DIVISION

5) $7 * 2 = 28$

6) $4 * 7 = \underline{\quad}$

7) $28 / 4 = \underline{\quad}$

8) $\underline{\quad} / 7 = 4$

PART V. NUMBER SENTENCES

Solve each number sentence below.

1) $3 * 3 + 6 = \underline{\quad}$

3) $6 - 2 + 0 = \underline{\quad}$

2) $6 / 3 * 2 = \underline{\quad}$

Circle the correct answer for each number sentence

1) $3 + 4 * 2 =$ a) 11 b) 14

2) $6 / 3 * 2 =$ a) 1 b) 4

3) $3 - 2 * 1 =$ a) 0 b) 1

4) $(3 + 3) / 3 =$ a) 2 b) 4

MATH TRAINING BOOKLET

HOW TO BE THE BEST AT WEST

Soon you will have the chance to play a brand new computer game called WEST. The idea of the game is a "race" between a train and a stagecoach along a path from 0 to 70 that contains TOWNS and SHORTCUTS. The first one to reach HOME (70) is the winner. You will be playing against the computer. To make your moves along the path you tell the computer that you want to take a turn and the machine will "spin" three spinners on the screen to get three numbers to work with. These numbers are used in a number sentence and the result determines how far you will move.

Look at the sample screen from the WEST game on the next page. Notice the three spinners and the path from 0 to 70. You will get all the game directions when you get to the computer. But, first -- let's go over some basic information you'll need to understand.

First of all, WEST has a special rule for forming number sentences:

you cannot repeat an operation or use a number twice in the same number sentence.

Secondly, WEST uses 2 important symbols:

a slash (/) for division and a star (*) for multiplication. For example, to show

"4 divided by 2 times 1 equals 2"

you would write 4 / 2 * 1 = 2

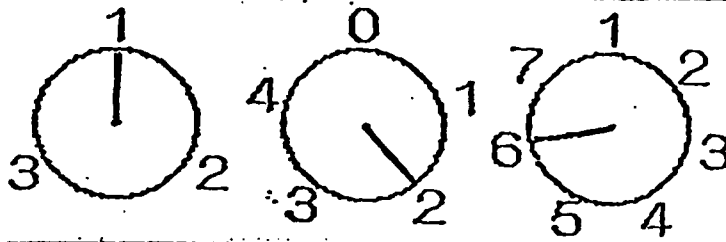
And finally, you may not use fractions for answers.

After you look at the game board on page 2,

TURN TO PAGE # 3-->

SAMPLE OF WEST GAME

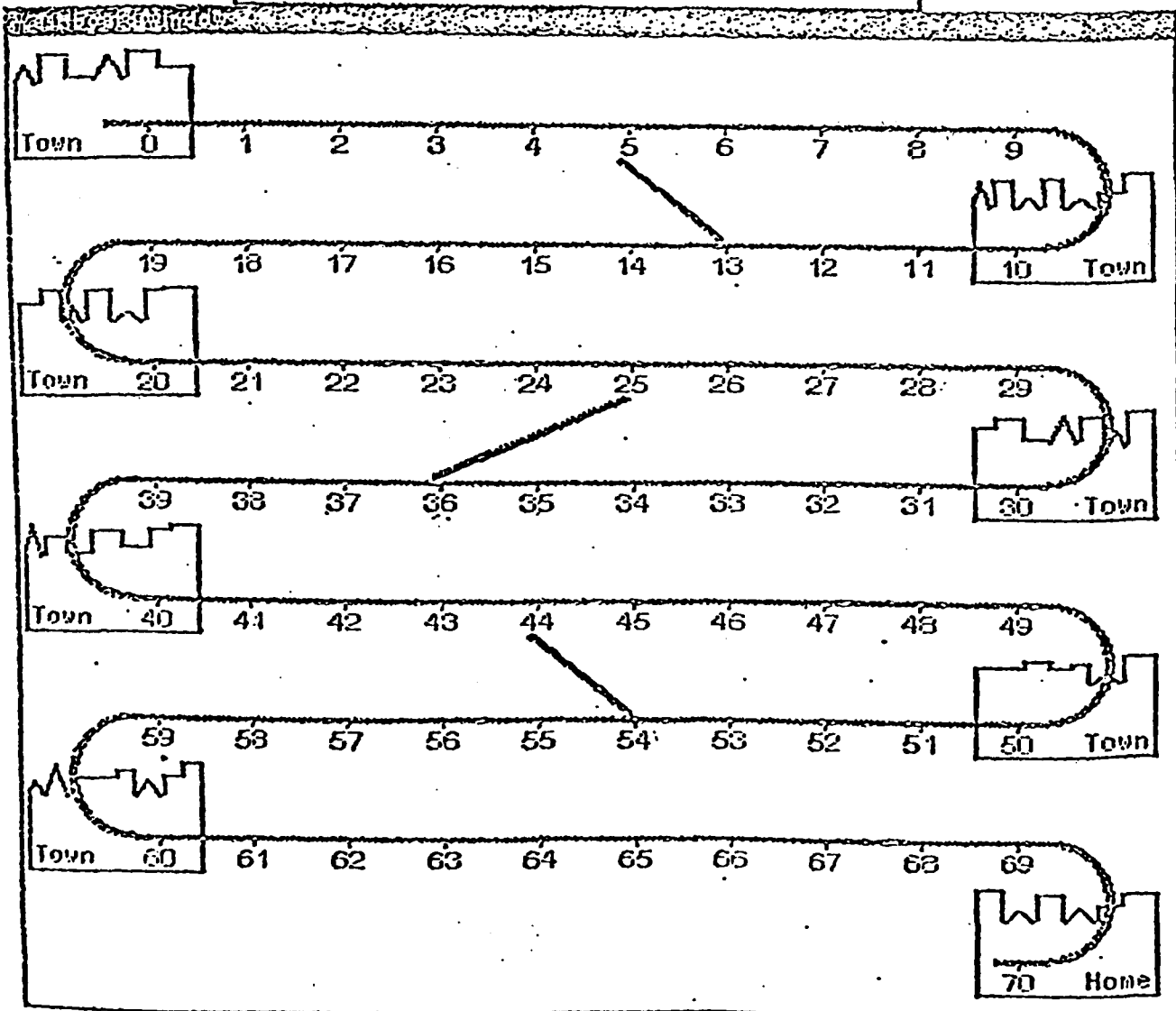
Stagecoach's turn



The numbers are: 1 2 6

<input type="text"/>	<input type="text"/>	<input type="text"/>	+	-	*	/	Hint
<input type="text"/>	<input type="text"/>	<input type="text"/>	(bs	ok	

YOUR MOVE:



In order to play WEST well, you will need to think about some important math ideas. This booklet will help you get ready! Read each page and answer the questions carefully. Work through all the pages at your own pace. Raise your hand when you are done.

READY TO BEGIN?

OK! TURN THE PAGE AND GET STARTED.

Math Idea # 1 - YOU MUST USE CORRECT ORDER OF OPERATIONS!

Let's say your 3 spinner numbers are

2 4 5

If you write

2 + 4 - 5

what number of spaces will you move?

Example of Coaching from WEST

=>

YES	NO
-----	----



























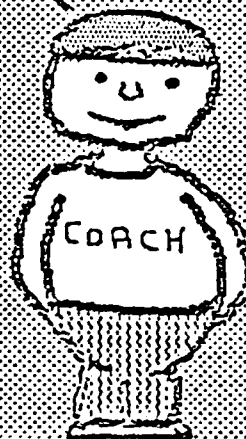
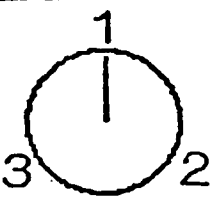
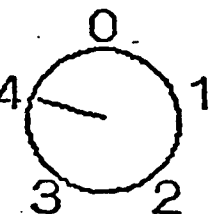
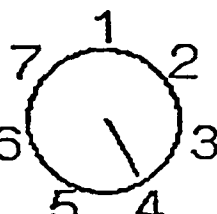
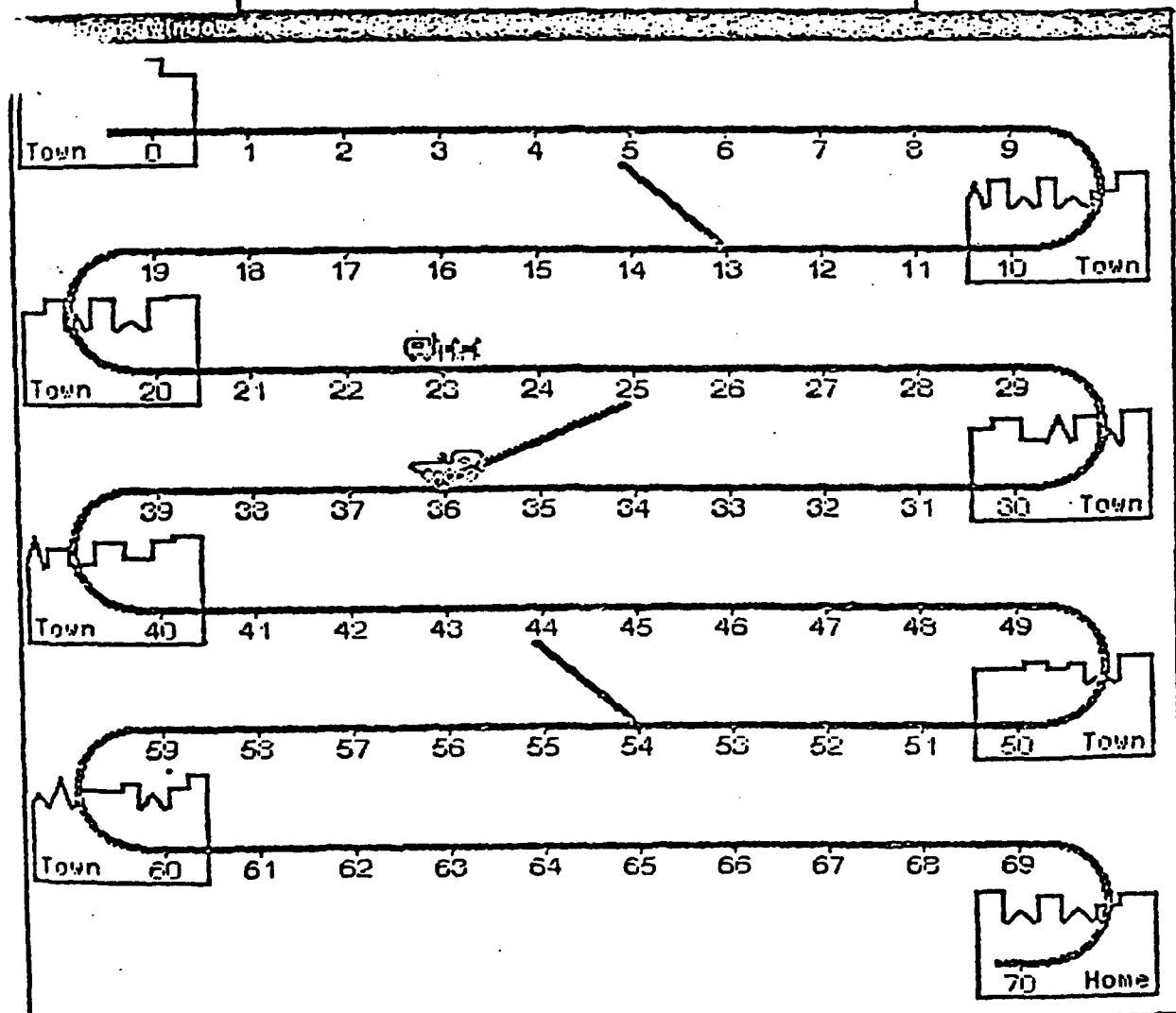


Figure 8
WEST Board Game

Stagecoach's turn			
			
The numbers are: 1 4 4			
<div>1</div>	<div>4</div>	<div>4</div>	<div>+</div> <div>-</div> <div>*</div> <div>/</div> <div>(</div> <div>)</div> <div>bs</div> <div>ok</div> <div>Hint</div>
YOUR MOVE:			



If you wrote " 1 ", you're right!

Let's say your spinners are

3 2 6

If you write

2 * 6 / 3

what number of spaces will you move?

Did you write " 4 "? Good work!

Suppose you have spinners

1 3 1

If you write

1 + 1 * 3

how many spaces will you move?

OOPS! Did you write " 6 "?

Actually the correct answer is 4 !

Here's why:

In math there are rules about which operations are done first in a number sentence.

Here are the rules to follow:

- 1) always add & subtract from left to right

Examples $7 - 3 + 1 = 5$

$$2 + 3 - 4 = 1$$

- 2) always multiply & divide from left to right

Examples $2 * 6 / 3 = 4$

$$4 / 2 * 5 = 10$$

- 3) always multiply or divide before you add or subtract

Examples $1 + 1 * 3 = 6$ (multiply first!)

$$5 - 4 / 2 = 3$$
 (divide first!)

Try this!

Your spinners are 4 0 5

and you write $5 - 0 * 4$. How many spaces will you move _____?

The correct answer is 5!

NOW!

Suppose your spinners are

1 2 6

If you write

$(1 + 2) \times 6$

how many spaces will you move?

Did you write " 18 "?

Well, the answer is 18 because of the next math idea!

Math Idea # 2 YOU CAN USE PARENTHESES IN NUMBER SENTENCES!

Numbers in parentheses are always operated on first. For example,

$$(3 - 1) * 3 = 6$$

Put () in this number sentence to make it correct:

$$4 + 2 * 2 = 12$$

$$= 12$$

Did you write $(4 + 2) * 2 = 12$?

The parentheses tell you to do addition first!

Try this one!

Your spinner numbers are

1 2 5

Solve each equation below:

#1 $5 - 1 + 2 =$

#2 $5 - (1 + 2) =$

#1 $5 - 1 + 2 = 6$ because you SUBTRACT first!

#2 $5 - (1 + 2) = 2$ because you ADD first!

See how parentheses change the answer!!

Try one more!

Suppose your spinners are

3 0 4

Make each number sentence correct with parentheses:

$$3 - 0 * 4 = 12$$

$$0 * 4 + 3 = 0$$

The correct answers are:

$$(3 - 0) * 4 = 12$$

$$0 * (4 + 3) = 0$$

Order of operations and parentheses are important to know!

Here's another important math idea:-->

Math Idea #3 SOME NUMBER SENTENCES HAVE NEGATIVE ANSWERS.

Look at this example:

Suppose your spinners are

2 1 4

If you write

2 - 1 + 4

what is your move?

Did you get 5 ? Good!

But suppose you write

2 - (1 + 4)

What is your move?

$$2 - (1 + 4) = -3 ! \quad \text{You move } \underline{\text{back } 3}!$$

When a large number is subtracted from a smaller number, we call the answer "negative". This answer is

"negative 3"

How about this one:

Your spinners are:

3 0 5

If you write

3 * 0 - 5

what is your move?

$$3 * 0 - 5 = -5$$

Since your answer is **negative 5**

you move back 5!

Solve each number sentence below:

$$2 + 0 - 3 = \underline{\quad}$$

$$4 / 2 - 5 = \underline{\quad}$$

$$5 - (3 * 3) = \underline{\quad}$$

Here are the correct answers:

$$2 + 0 - 3 = -1 \quad (\text{negative } 1)$$

$$4 / 2 - 5 = -3 \quad (\text{negative } 3)$$

$$5 - (3 * 3) = -4 \quad (\text{negative } 4)$$

Now look at the last important math idea!

Math Idea #4 SOME NUMBER SENTENCES FOLLOW PATTERNS.

For example,

If you have spinners 2 1 6 and you write

$$(1 + 2) * 6$$

what is your move?

Right! $(1 + 2) * 6 = \underline{18}$

18 is the largest possible answer you can get with 2 1 6.

How about this:

Your spinners are 1 4 3

You write $(3 + 1) * 4.$

Your move is _____

$(3 + 1) * 4 = 16.$ You move 16 spaces.

16 is the largest possible answer you can get with

1 4 3

One more!

Your spinners are 2 3 5

You write $(3 + 2) * 5$

Your move is _____

$$(3 + 2) * 5 = 25$$

Can you get a larger number than 25? _____

No! 25 is the largest number you can get with

2 3 5

You can always get the largest answer by using a special PATTERN.

Add the two smaller numbers within a set of parentheses and then multiply this quantity by the largest of the three spinner numbers.

For example - look at these again!

$$1 \ 6 \ 2 \quad (1 + 2) * 6 = 18$$

$$4 \ 3 \ 1 \quad (3 + 1) * 4 = 16$$

$$3 \ 2 \ 5 \quad (3 + 2) * 5 = 25$$

Make each number sentence below correct by using parentheses:

$$3 + 2 * 4 = 20$$

$$6 * 1 + 4 = 30$$

$$5 + 1 * 7 = 42$$

The correct number sentences are:

$$(3 + 2) * 4 = 20$$

$$6 * (1 + 4) = 30$$

$$(5 + 1) * 7 = 42$$

Remember -- smaller numbers are added in the parentheses and then multiplied by the largest number!

Now you have practiced 4 important math ideas:

- o using rules for order of operations
- o using parentheses
- o getting negative answers
- o using the pattern for largest answer

See how well you remember them as you work on the following problems!

Solve these number sentences:

1) $4 + 3 * 2 = \underline{\hspace{2cm}}$

2) $(7 - 1) / 2 = \underline{\hspace{2cm}}$

3) $3 - (2 + 5) = \underline{\hspace{2cm}}$

4) $6 * (5 + 1) = \underline{\hspace{2cm}}$

5) $4 / 2 * 2 = \underline{\hspace{2cm}}$

Can you make 10 using 5 2 4 ?

$\underline{\hspace{2cm}} = 10$

Can you get 0 using 3 0 6 ?

$\underline{\hspace{2cm}} = 0$

RAISE YOUR HAND WHEN YOU ARE FINISHED. YOUR ANSWERS WILL BE CHECKED FOR YOU.

HAVE FUN PLAYING WEST!

MATH AND STRATEGY TRAINING
BOOKLET

HOW TO BE THE BEST AT WEST

Soon you will have the chance to play a brand new computer game called WEST. The idea of the game is a "race" between a train and a stagecoach along a path from 0 to 70 that contains TOWNS and SHORTCUTS. The first one to reach HOME (70) is the winner. You will be playing against the computer. To make your moves along the path you tell the computer that you want to take a turn and the machine will "spin" three spinners on the screen to get three numbers to work with. These numbers are used in a number sentence and the result determines how far you will move.

Look at the sample screen from the WEST game on the next page. Notice the three spinners and the path from 0 to 70. You will get all the game directions when you get to the computer. But, first -- let's go over some basic information you'll need to understand.

First of all, WEST has a special rule for forming number sentences:

you cannot repeat an operation or use a number twice in the same number sentence.

Secondly, WEST uses 2 important symbols:

a slash (/) for division and a star (*) for multiplication. For example, to show

"4 divided by 2 times 1 equals 2"

you would write 4 / 2 * 1 = 2

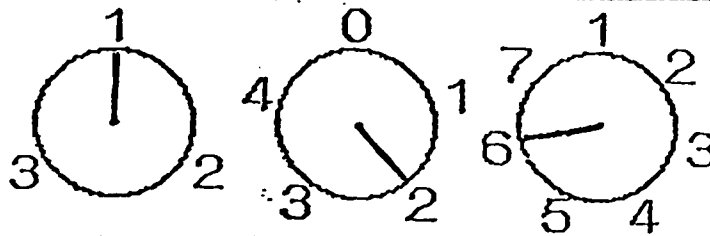
And finally, you may not use fractions for answers.

After you look at the game board on page 2,

TURN TO PAGE # 3-->

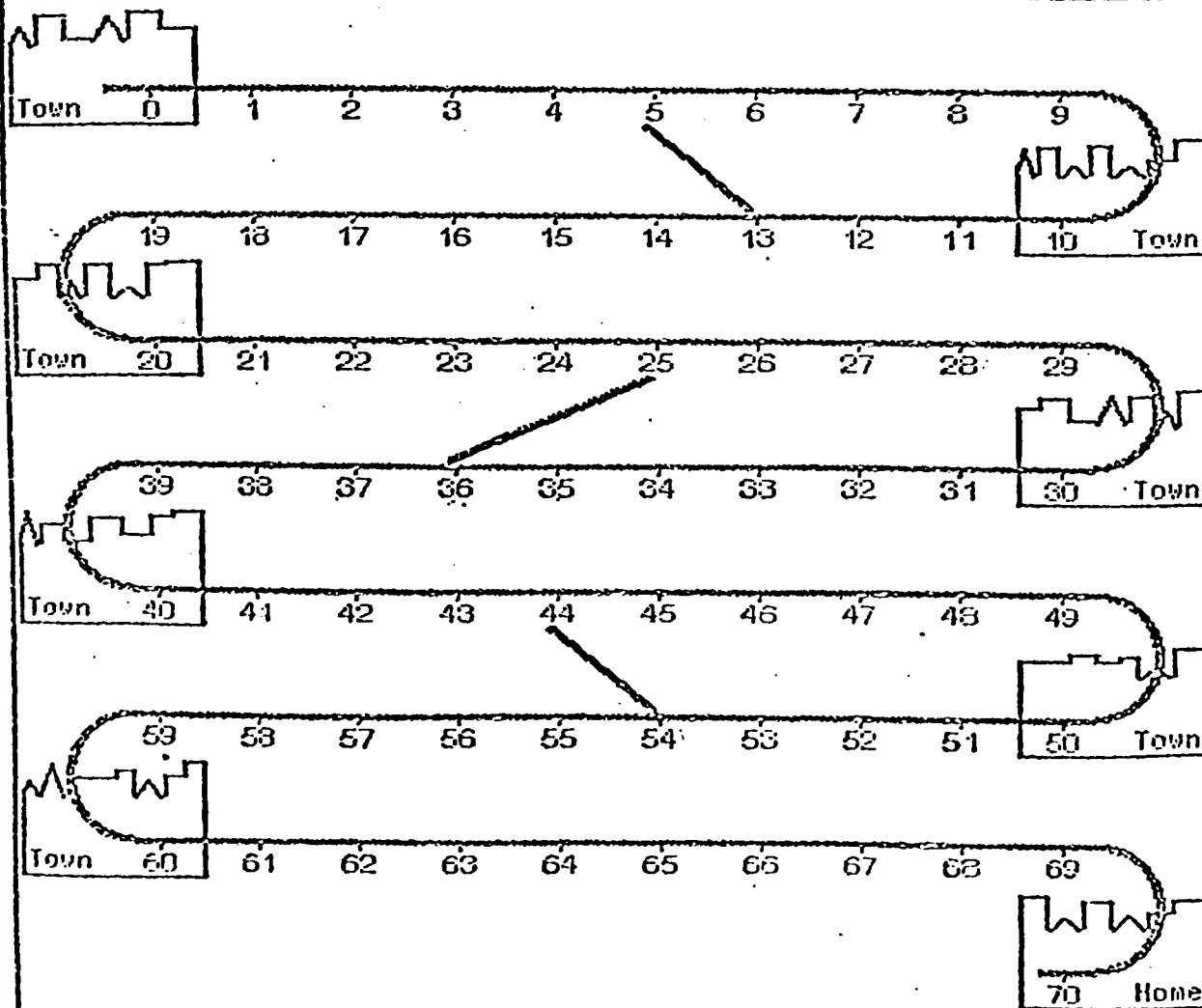
SAMPLE OF WEST GAME

Stagecoach's turn



The numbers are: 1 2 6

YOUR MOVE:



In order to play WEST well, you will need to think about some important math and game ideas. This booklet will help you get ready! Read each page and answer the questions carefully. Work through all the pages at your own pace. Raise your hand when you are done.

Ready to begin? OK! Turn the page and GET STARTED!

GAME IDEA #1 - LOOK FOR YOUR OPTIONS

When it is your turn, ask yourself -

** What are my options? **

WEST has 4 special options! You can:

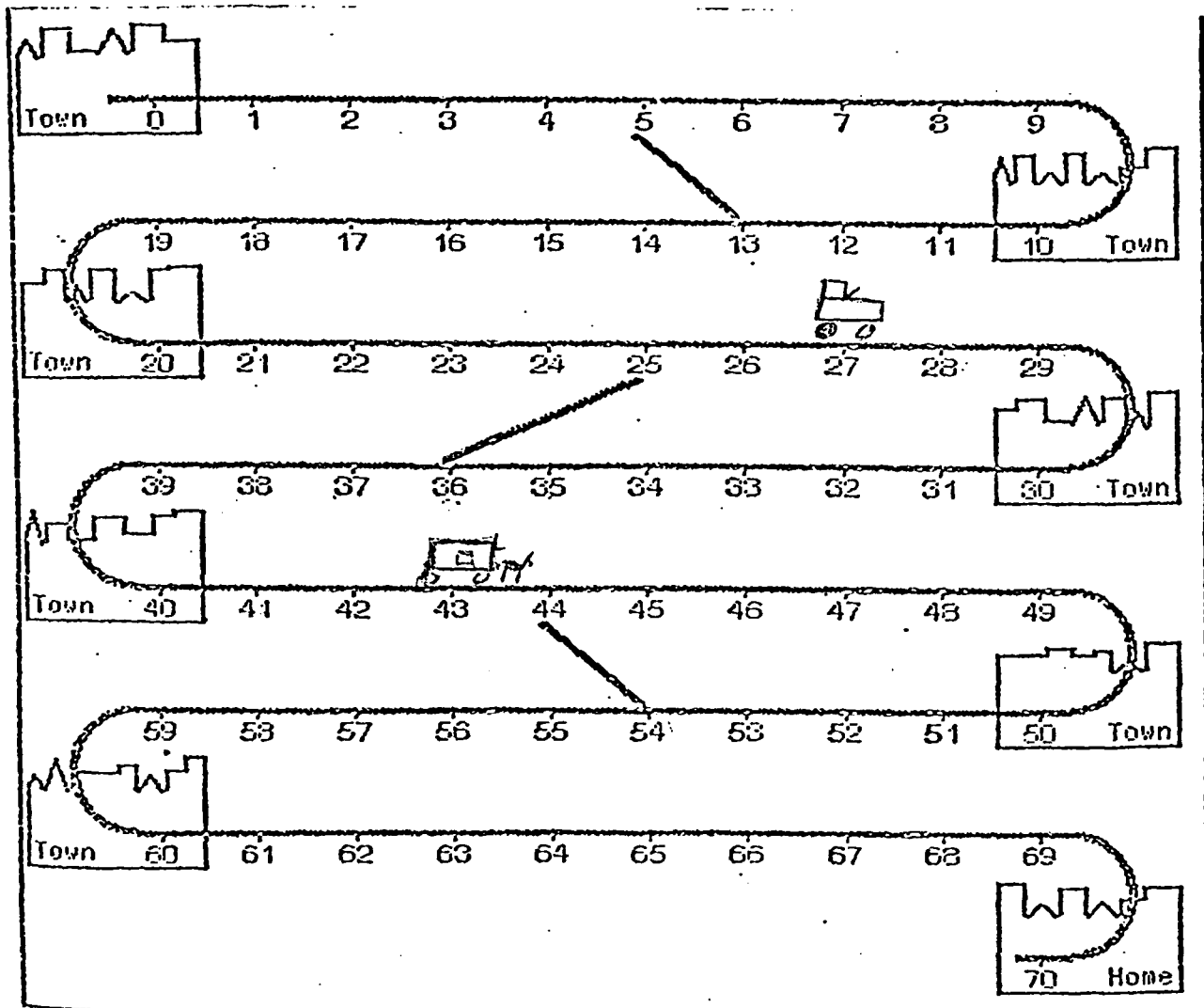
- 1) Go for a TOWN (because you can jump to the next TOWN)
- 2) Get on a SHORTCUT (because you can move to the next row)
- 3) Land on your opponent for a BUMP (because he/she must go back 2 TOWNS)
- 4) Move a BIG DISTANCE (because you will get far on the path)

How many special options does WEST give you? _____

You have 4 options to choose from -

TOWNS, SHORTCUTS, BUMPS, DISTANCE

Look at this game board -



Suppose you are the stagecoach. Which options would you consider?

- a) a TOWN, a BUMP, a SHORTCUT
- b) a TOWN, BIG DISTANCE, a BUMP
- c) a SHORTCUT, a TOWN, BIG DISTANCE

The answer that you should consider is C -

a SHORTCUT, a TOWN, BIG DISTANCE

Sometimes, only some of the options are good to consider!

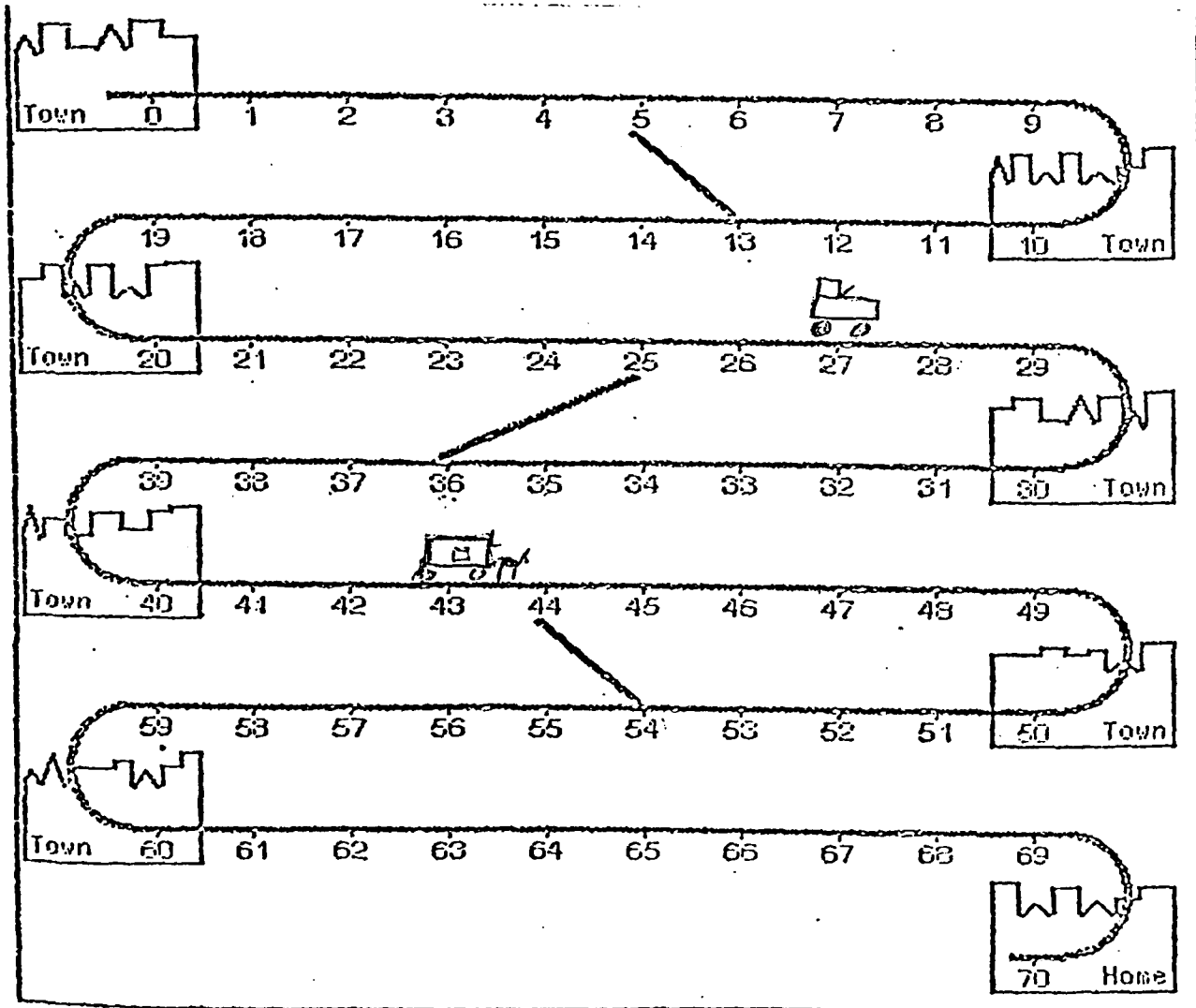
Once you think about options, you are ready for the next idea!

GAME IDEA #2 - FIGURE OUT HOW MANY SPACES YOU NEED FOR EACH OPTION

For each option, ask yourself -

**** How many spaces will I need to move? ****

Look at this gameboard again -



How many spaces does the train need to move to BUMP? _____

How many spaces does the stagecoach need to WIN? _____

The train needs 16 to BUMP!

The stagecoach needs 27 to WIN!

NOW! The next idea will tell you how to get the numbers you want!

GAME IDEA #3 - USE MATH RULES TO FIND WHICH NUMBERS ARE POSSIBLE

When you know what numbers you want, ask yourself,

** Which numbers are possible with the spinners I have? **

Suppose you would like to have either 7, 18, or 30.

Your spinners are 2, 3, 6.

Is it possible to get 7? _____

If so, how? _____ = 7

(DO NOT REPEAT NUMBERS OR OPERATIONS)

Yes, 7 is possible. You can use either

$$3 - 2 + 6 = 7 \quad \text{or} \quad 6 + 3 - 2 = 7.$$

Can you get 18 using 2, 3, 6? _____

If so, how? _____ = _____

Yes 18 is possible -

$$(3 + 6) * 2 = 18$$

Are you wondering -

- how will I know what number sentences to write??

Well, the next few pages will help you understand some important math rules for number sentences!

Let's say your spinners are

3 2 6

If you write

2 * 6 / 3

what number of spaces will you move?

Did you write "4"? Good work!

Suppose you have spinners

1 3 1

If you write

$1 + 1 * 3$

how many spaces will you move?

OOPS! Did you write "6"?

Actually the correct answer is 4 !

Here's why:

In math there are rules about which operations are done first in a sentence.

Here are the rules to follow:

- 1) always add & subtract from the left to right

Examples $7 - 3 + 1 = 5$

$$2 + 3 - 4 = 1$$

- 2) always multiply & divide from left to right

Examples $2 * 6 / 3 = 4$

$$4 / 2 * 5 = 10$$

- 3) always multiply or divide before you add or subtract

Examples $1 + 1 * 3 = 4$ (multiply first!)

$$5 - 4 / 2 = 3 \text{ (divide first!)}$$

NOW!

Suppose your spinners are

1 2 6

If you write

$(1 + 2) * 6$

how many spaces will you move?

Did you write "18"?

Numbers in parentheses are always operated on first. For example,

$$(3 - 1) * 3 = 6$$

Put () in this number sentence to make it correct:

$$4 + 2 * 2 = 12$$

$$= 12$$

Did you write $(4 + 2) * 2 = 12$?

The parentheses tell you to do addition first!

Try this one!

Your spinner numbers are

1 2 5

Solve each equation below:

#1 $5 - 1 + 2 =$

#2 $5 - (1 + 2) =$

In #1, $5 - 1 + 2 = 6$, you subtract first!

In #2, $5 - (1 + 2) = 2$, you add first!

That's the MATH - Here's the STRATEGY:

To get lots of different answers, try lots of different combinations with your spinner number.

Using the math rules, fill in the operations below to make the sentences correct.

#1 $(3 \square 1) * 4 = 16$

#2 $4 / (3 \square 1) = 2$

#3 $1 + 4 \square 3 = 13$

$$\#1 \quad (3 \boxed{+} 1) * 4 = 16$$

$$\#2 \quad 4 / (3 \boxed{-} 1) = 2$$

$$\#3 \quad 1 + 4 \boxed{*} 3 = 13$$

Okay - now try a game situation!

Read this carefully -

Kim wants 3 for a SHORTCUT, 9 for a TOWN, or just the BIGGEST DISTANCE. Her spinners are 3 1 6.

Help Kim out - write a number sentence for each option.

$$\#1 \quad \underline{\hspace{2cm}} = 3$$

$$\#2 \quad \underline{\hspace{2cm}} = 9$$

$$\#3 \quad \underline{\hspace{2cm}} = \underline{\hspace{1cm}} \text{ (BIGGEST MOVE POSSIBLE)}$$

#1 any of these is okay!

$$6 - 3 * 1 = 3$$

$$6 - 3 / 1 = 3$$

$$6 / (3 - 1) = 3$$

#2 sentences like these are correct!

$$(6 + 3) * 1 = 9$$

$$6 + 3 * 1 = 9$$

$$6 * 1 + 3 = 9$$

#3 the largest possible move is 24.

$$(1 + 3) * 6 = 24$$

You always get the largest answer when you add the two smaller number within a set of parentheses and then multiply the result by the largest number!

Now let's find out which move is BEST!

GAME IDEA #4 - IF YOU CAN'T WIN , GET THE BEST DISTANCE
POSSIBLE BETWEEN YOU AND YOUR OPPONENT.

When you have a few moves to choose from, ask yourself:

** Which move will put me the BEST distance from my opponent? **

Always look at the DISTANCE BETWEEN you and the opponent.

For example, if one option puts you 16 spaces ahead and another one puts you 7 spaces ahead of your opponent, use the first option and get 16 spaces away from your opponent!

Let's look at the 4 Game Ideas again -

1 - Look for your options

2 - Figure out how many spaces you need for each option

3 - Use math rules to find which numbers are possible

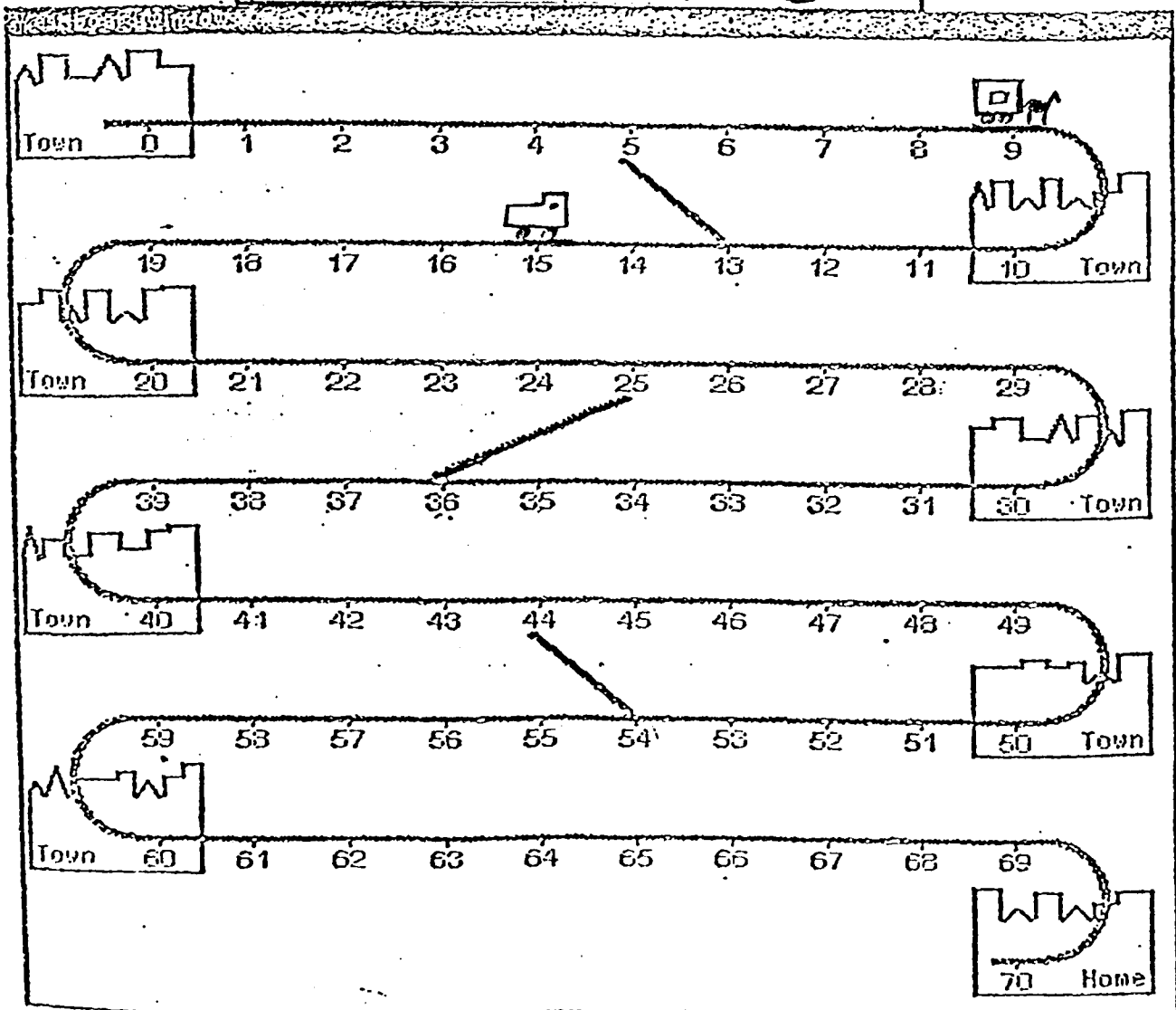
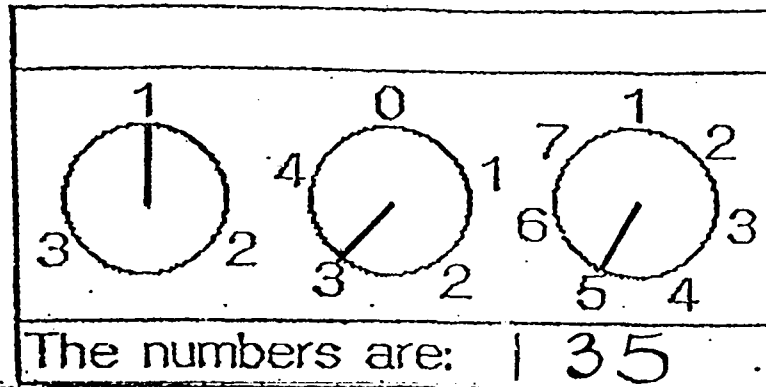
4 - Either WIN or get the BEST DISTANCE BETWEEN you and your opponent

Use the sample game on page 23 as you answer these questions.

1. The stagecoach needs how many to BUMP? _____
2. The stagecoach needs how many to reach a SHORTCUT? _____
3. The BIGGEST DISTANCE the stagecoach can move is _____.
4. If the stagecoach had either a 1 or 11, what special option could it use? _____
5. Circle which moves are possible for the stagecoach.

BUMP SHORTCUT BIGGEST DISTANCE TOWN
6. Which is BEST? _____

YOU ARE THE STAGECOACH!



EXPERIMENTER'S GAME NOTES

"We are studying how students can learn to play a math game called WEST on a special kind of computer---an intelligent one.

FOR CONTROL Ss ONLY:

"Soon you will have a chance to play the game, but first I'd like to ask you some questions about games, computers, & math."

GIVE PRE-GAME INTERVIEW. PUT INTERVIEW PAGE IN S's FOLDER.

"Now we will give you a very short math paper to do. You will have 5 min. to finish as much as you can. Do your best, and raise your hand if you have a question."

GIVE PRETEST. ALLOW 5 MIN. TOTAL. TELL Ss WHEN 1 MIN. REMAINS. WHEN Ss ASK QUESTIONS, REFER THEM TO APPROPRIATE DIRECTIONS IN BOOKLET. AVOID GIVING SUBSTANTIVE ANSWERS. PUT FINISHED PRETESTS IN Ss' FOLDERS.

FOR ALL Ss:

"Now you will get to play WEST with the computer. First I will explain some things you need to know about how to play the game, and we'll play a sample game so you can see how it's done. You will have about 40 min. to play, so I may have to stop you in the middle of a game when the time is up." ...

GIVE DIRECTIONS AND LET SUBJECT PLAY GAME. STOP GAME AFTER 40 MIN OF PLAY.

OBSERVATIONAL RATINGS

NAME _____

- | | NOT AT
ALL
1 | 2 | SOME
3 | 4 | A LOT
5 |
|---|---------------------------|---|-----------|---|--------------------------|
| 1. How enthusiastic was student? | 1 | 2 | 3 | 4 | 5 |
| 2. How much effort did student display? | 1 | 2 | 3 | 4 | 5 |
| 3. How often did student comment?
TALLY: | 1 | 2 | 3 | 4 | 5 |
| 4. How often did student ask questions?
TALLY:
RECORD QUESTIONS BELOW. | 1 | 2 | 3 | 4 | 5 |
| 5. Did student's playing ability improve? | 1 | 2 | 3 | 4 | 5 |
| 6. How did the student react to losing? | | | | | |
| | being
discouraged
1 | 2 | 3 | 4 | being
challenged
5 |
| 7. How would you describe the student's general playing strategy? | | | | | |
| | reflective
1 | 2 | 3 | 4 | impulsive
5 |
| 8. In general, what type of math ability did the student display? | | | | | |
| | poor
1 | 2 | 3 | 4 | excellent
5 |
| 9. In general, what type of game playing strategy did the student display? | | | | | |
| | poor
1 | 2 | 3 | 4 | excellent
5 |
| 10. Did the student mention training? (If yes, circle below or comment)
(e.g. order of operations, parentheses, formula for largest number, using
negative number to move backwards, maximum delta) | | | | | |

ADDITIONAL COMMENTS

EXPERIMENTER'S POSTTESTING NOTES

"Today we would like you to do some math and game problems that are related to the WEST game you played yesterday on the computer. You'll do the math section first. You have 20 minutes, but it may not take you that long. If you have a question, raise your hand. Do your best, but if you can't figure out a problem, go on to the next one, and you can come back to the hard one later."

PASS OUT MATH POST TEST. ALLOW 20 MINUTES. TELL Ss WHEN 5 MIN REMAIN. PUT FINISHED TESTS IN FOLDERS.

"Here are some game problems. You will have about 25 minutes to finish, but it may not take you that long. As before, do the best you can, and raise your hand if you have a question."

PASS OUT STRATEGY POST TEST. ALLOW 25 MIN. TELL Ss WHEN 5 MIN REMAIN. PUT FINISHED TESTS IN FOLDERS.

NOTE: A number of Ss were confused by the strategy post test format of 1 page of questions that pertained to the following page illustrating a game of WEST with the spinners and positions of the stagecoach and train shown. In this case, E explained that the two pages of similar color went together and that the second page illustrated the game referred to on the previous page.

"Now, just one more thing before you return to class. I'd like to ask you some questions about what you have been doing."

GIVE FINAL INTERVIEW. PUT IN FOLDER. THANK S FOR PARTICIPATING AND ESCORT BACK TO CLASS.

MATHEMATICS POSTTEST

PART I. SOLVING NUMBER SENTENCES

DIRECTIONS: Solve each number sentence below. Put the answers on the lines provided.

$$1) 7 * 0 + 3 = \underline{\quad\quad} \qquad 7) 4 - 2 * 2 = \underline{\quad\quad}$$

$$2) (2 + 6) / 4 = \underline{\quad\quad} \qquad 8) 3 + 0 - 5 = \underline{\quad\quad}$$

$$3) 6 * (4 + 3) = \underline{\quad\quad} \qquad 9) 7 / (4 - 3) = \underline{\quad\quad}$$

$$4) 4 + 6 / 2 = \underline{\quad\quad} \qquad 10) 7 - 4 + 2 = \underline{\quad\quad}$$

$$5) 0 * (3 / 3) = \underline{\quad\quad} \qquad 11) 2 \times 6 / 4 = \underline{\quad\quad}$$

$$6) 2 / 2 - 4 = \underline{\quad\quad} \qquad 12) 4 * (1 + 2) = \underline{\quad\quad}$$

POSTTEST

ALL WEST RULES APPLY:

- do not repeat operations
- use each number once
- no fraction answers

PART II. WRITING NUMBER SENTENCES

DIRECTIONS: In each section below you will be writing number sentences.

NOTE: Change the order of the numbers and you may use parentheses as needed.

A. Write a number sentence with each set of numbers below that equals the LARGEST possible answer:

- 1) 1, 2, and 5 _____
- 2) 2, 4, and 3 _____
- 3) 3, 7, and 1 _____

B. Write a number sentence for each problem below.

- 1) Use 2, 2 and 7 to equal 12 _____
- 2) Use 1, 4 and 6 to equal -2 _____
- 3) Use 2, 4 and 5 to equal 10 _____
- 4) Use 6, 3 and 1 to equal 15 _____
- 5) Use 2, 6 and 4 to equal 1 _____

POSTTEST

PART III. MAKING DIFFERENT NUMBER SENTENCES

DIRECTIONS: Use 2, 1 and 6 in number sentences to get as many different answers as you can.

Use only the numbers 2, 1 and 6.

Use the WEST Rules!

PART IV. EXTENDING YOUR SKILLS

DIRECTIONS: Solve the number sentences below:

1) $15 * 3 - 48 =$ _____

$$2) \quad (27 + 28) / 5 =$$

PART V. USING LARGER NUMBERS

DIRECTIONS: Answer each question below:

1) What is the largest possible number you can make using

$$\begin{array}{ccc} 12 & 20 & 9 \\ & & = \end{array}$$

2) How can you get 32 using 10 14 8 ?

_____”

Strategy Posttest

This booklet contains 5 sample WEST moves. Each move has a page of questions about the stagecoach. Read the questions carefully and then write your answers.

All WEST rules apply!!

Strategy Posttest

Questions for Game 1

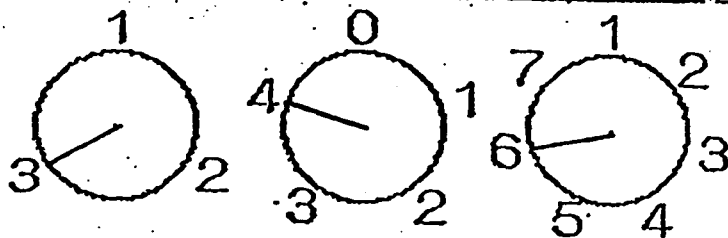
Circle the correct answer:

- 1) Which is a better move for the stagecoach?
a) 2 spaces b) 8 spaces
- 2) Which is a better move for the stagecoach?
a) 21 spaces b) 16 spaces
- 3) Which is a better move for the stagecoach?
a) 27 b) 22
- 4) Which is a better move for the stagecoach?
a) 42 b) 30

GAME 1

SELECTING THE MOVE THAT UTILIZES A SPECIAL OPTION

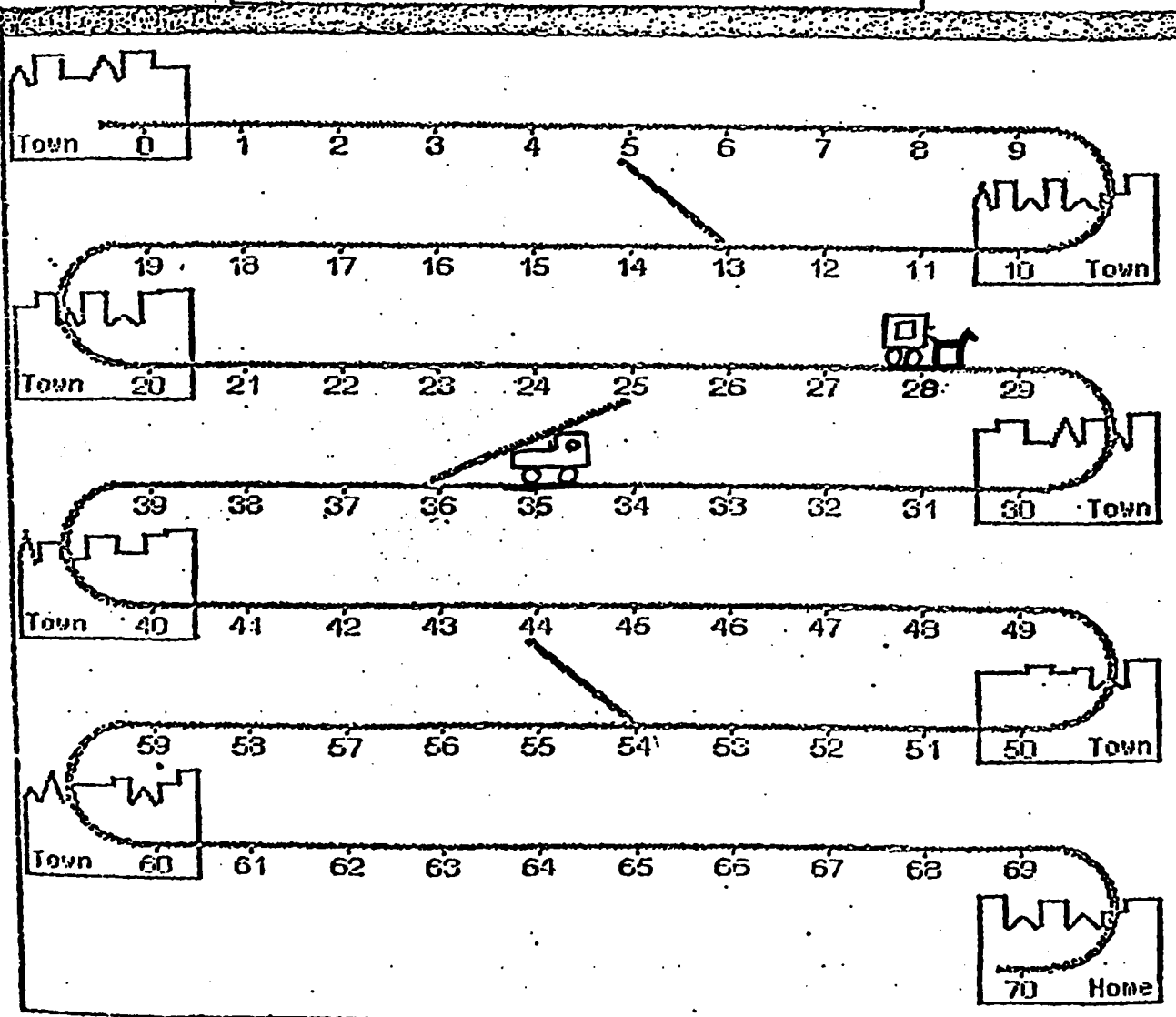
Stagecoach's turn



The numbers are:

3 4 6

YOUR MOVE:

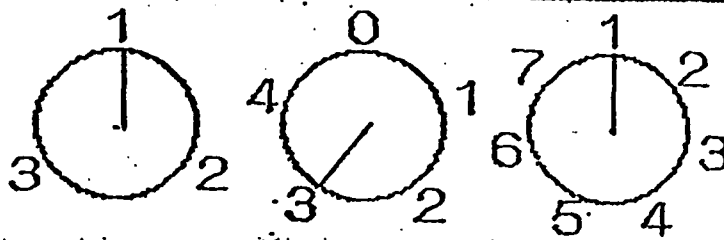


Questions for Game 2 - Circle the correct answer:

- What does the stagecoach need in order to BUMP the train?
a) 4 b) 5
- What does the stagecoach need to get to the SHORTCUT?
a) 14 b) 12
- What does the stagecoach need to get to TOWN 20?
a) 1 b) 9
- What does the stagecoach need to get to TOWN 10?
a) -1 b) 1

DETERMINING HOW MANY SPACES ARE REQUIRED

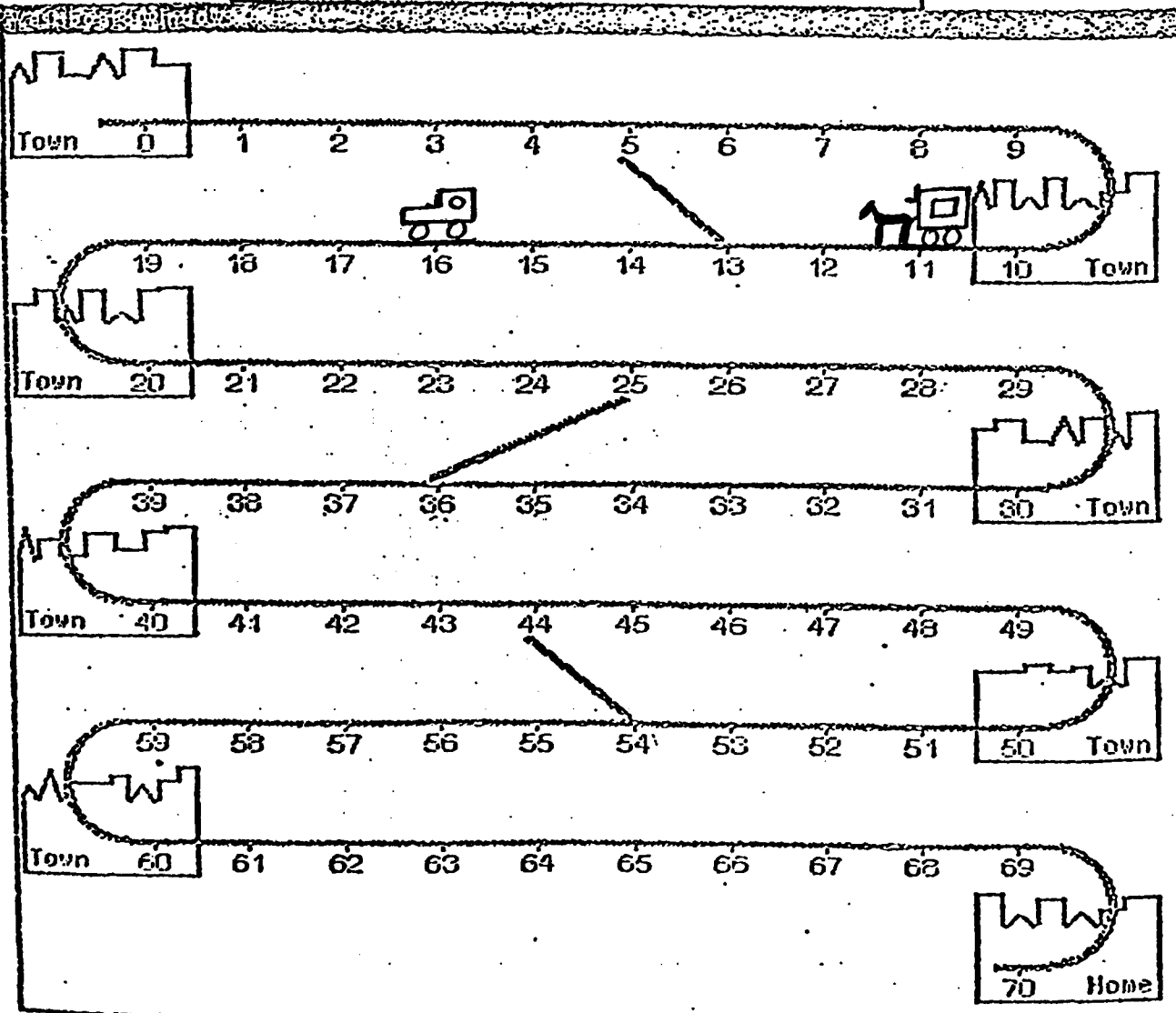
Stagecoach's turn



The numbers are:

1 3 1

YOUR MOVE:



Questions for Game 3

The stagecoach would like to make any of these moves:

3 4 8 14 30

Look at the spinners and decide which moves are possible.

Circle a) or b)

Show a number sentence for all possible moves

1) 3 is a) possible _____ = 3

b) impossible

2) 4 is a) possible _____ = 4

b) impossible

3) 8 is a) possible _____ = 8

b) impossible

4) 14 is a) possible _____ = 14

b) impossible

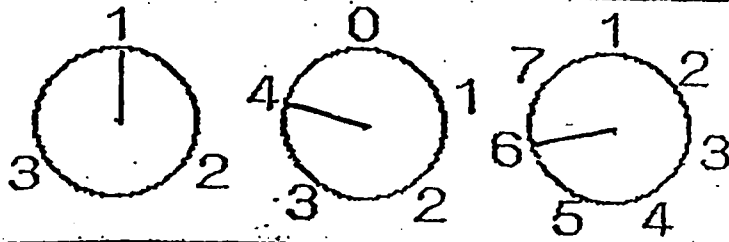
5) 30 is a) possible _____ = 30

b) impossible

GAME 3

DETERMINING POSSIBLE MOVES

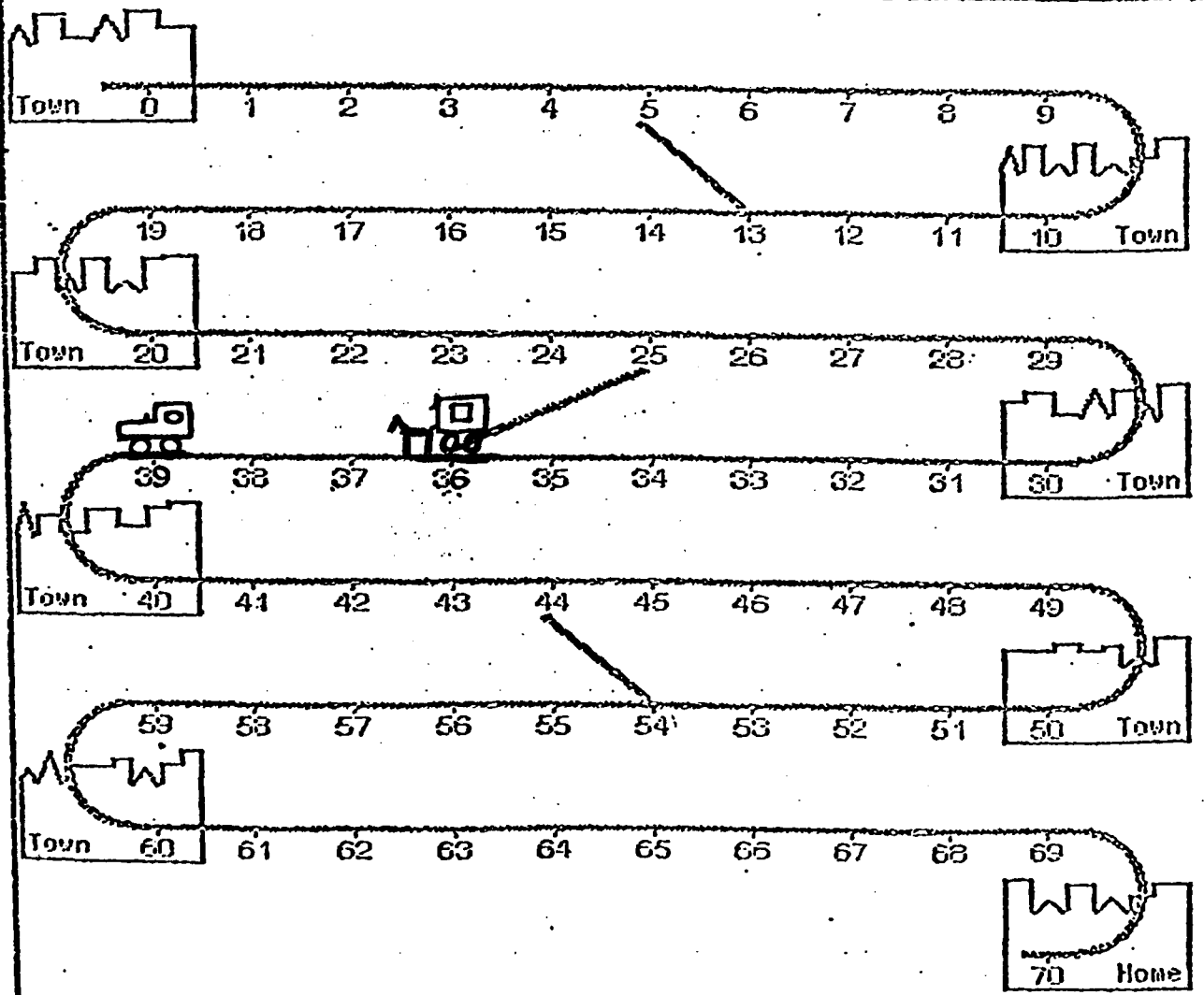
Stagecoach's turn



The numbers are:

1 4 6

YOUR MOVE:



Questions for Game 4

Fill in the blanks:

The stagecoach can make these moves:

3 4 8 12

1) Which is BEST? _____

2) What equation would you use for the BEST move you chose?

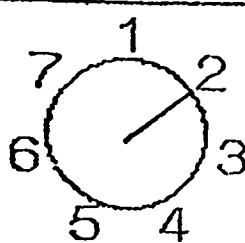
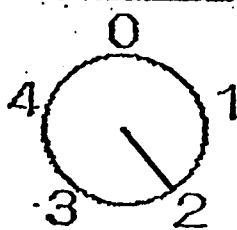
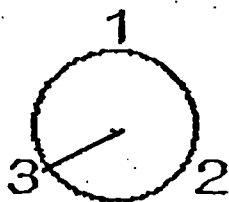
_____ = _____

3) This move is BEST because _____

GAME 4

SELECTING THE BEST MOVE

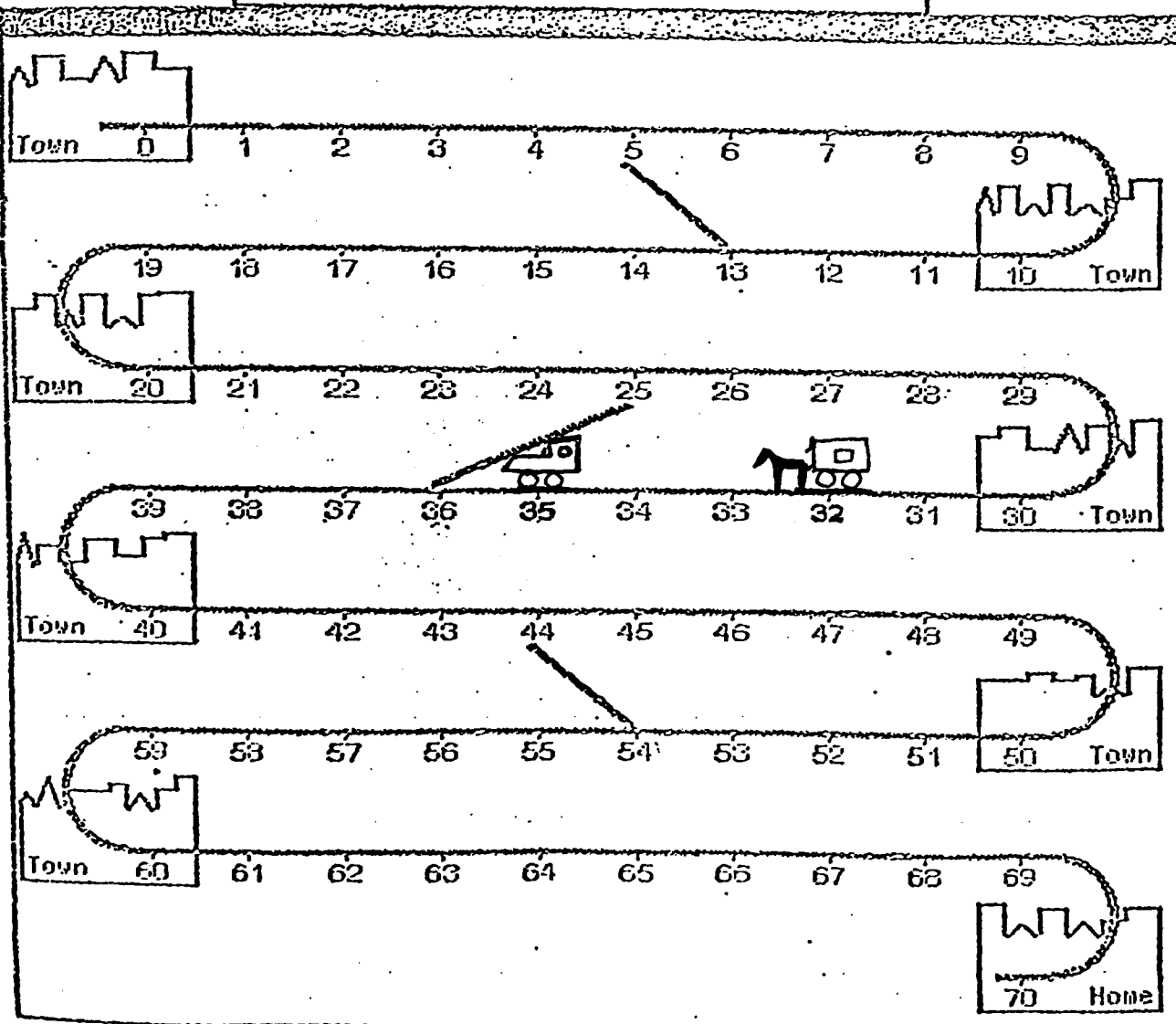
Stagecoach's turn



The numbers are:

3 2 2

YOUR MOVE:



Questions for Game 5

1) What options would you consider for the stagecoach? Circle a) or b)

a) BUMP, SHORTCUT, HOME

b) BUMP, TOWN, HOME

2) What numbers would you like to get?

Circle a) or b)

a) 4, 6, or 16

b) 4, 0, or 16

3) Circle all the numbers that are possible to get:

0 4 6 16

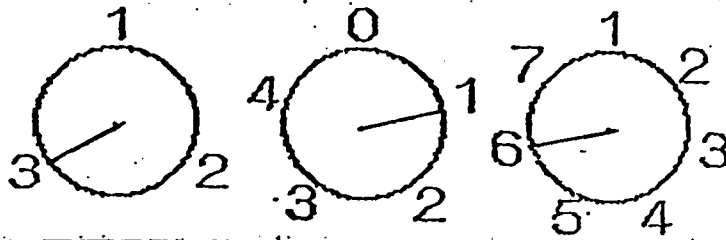
4) Which move is BEST? _____

Show the equation you would use to get it:

_____ = _____

GAME 5

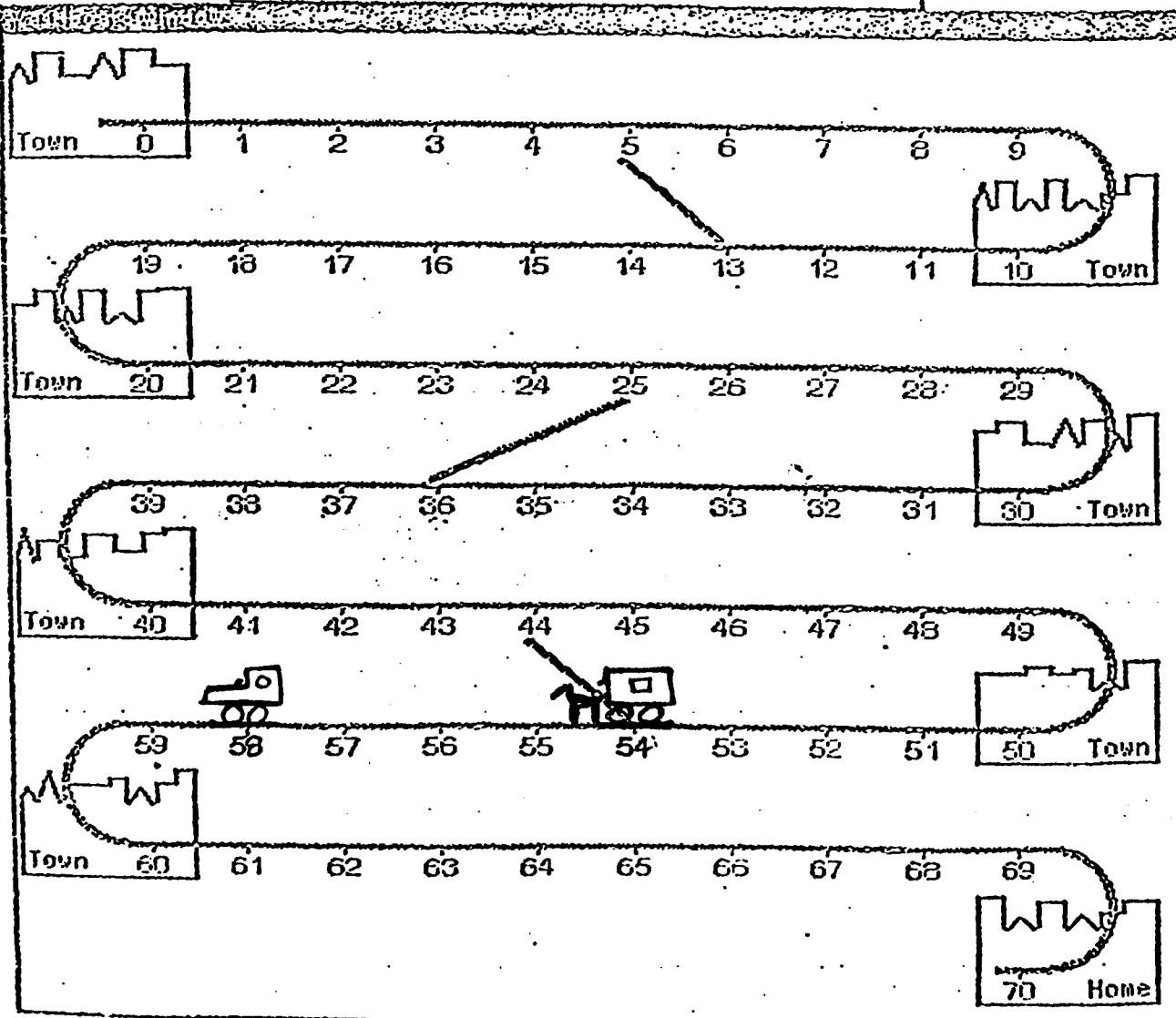
Stagecoach's turn



The numbers are:

3 1 6

YOUR MOVE:



This last page is about a game just like WEST. All the WEST rules apply. The only difference is that the game board is longer! This game is called BIG WEST.

The BIG WEST board goes up to 100. So there are TOWNS at 70, 80 and 90. HOME is at 100. Read the questions below and circle your answers!

- 1) If you are on 85, what two moves can get you HOME?
a) 5, 10 b) 5, 15 c) 15, 25

- 2) If you BUMP your opponent at space 95, where will he/she go?
a) 60 b) 70 c) 80

- 3) You are on 82, your opponent is on 89. Which move below is BEST:
a) you use 7 to BUMP b) you use 15 to get on 97.

NAME _____

POST-GAME INTERVIEW

1. What was the hardest part of the game for you?
2. What was the easiest part?
3. Did you learn anything from playing the game? What?
4. How did you like the coach? Did it help you? Did it bother you or interfere?
5. How did you like the booklet on math (and game playing skills)?
Was it confusing?
Was it boring?
Were you able to remember it by the time you got to play the game on the computer?
Was it helpful in playing the game? If so, what was most helpful?
6. Would you like to have had help on anything else? If so, with what?
7. What 3 things about the game would you tell a friend to help him/her out?

ADDITIONAL COMMENTS